A Single-Type Semantics for Natural Language

Liefke, K.L.

Publication date:
2014

Document Version
Publisher's PDF, also known as Version of record

This dissertation is a contribution to the logical and ontological foundations of natural language semantics. In particular, it provides a formal semantics which interprets a benchmark fragment of English through the use of a *single* type of semantic primitive. The possibility of such a *single-type semantics* has been conjectured by Barbara Partee (2006) to account for recent findings in language development. The presented semantics supports Partee's conjecture and accommodates its empirical motivation. The development of this semantics answers a number of questions about the salience of the Montagovian type system (cf. Montague, 1970a; 1973), the robustness of semantic theories with respect to their basic-type choice, and the classification of these theories according to their objects' informational strength.

Kristina Liefke (*1983) is a member of the Munich Center for Mathematical Philosophy (MCMP) at LMU Munich. She was trained in philosophy and English linguistics at CAU Kiel and at UCLA. Since 2009, she has been a PhD student at the Tilburg Center for Logic and Philosophy of Science (Tilburg University). She joined the MCMP in 2012.

ISBN 978-94-6259-131-8

Kristina Liefke

A Single-Type Semantics for Natural Language

A Single-Type Semantics for Natural Language

Kristina Liefke

A Single-Type Semantics for Natural Language

Kristina Liefke

Cover photo 'Antique Letterpress Wood Type Printers Block Letter O',
Courtesy Pamela Kaplan for *Preserve Cottage*

Printed and bound by Ipskamp Drukkers, Enschede, The Netherlands
ISBN 978-94-6259-131-8

A Single-Type Semantics for Natural Language

Een Enkele-Type-Semantiek voor de Natuurlijke Taal
(met een samenvatting in het Nederlands)

Proefschrift

ter verkrijging van de graad van doctor
aan Tilburg University
op gezag van de rector magnificus,
prof. dr. Ph. Eijlander,
in het openbaar te verdedigen ten overstaan van een
door het college voor promoties aangewezen commissie
in de aula van de Universiteit

op vrijdag 25 april 2014 om 10.15 uur

door

Kristina Liefke

geboren op 20 april 1983 te Neumünster, Duitsland

Promotiecommissie

Promotor: prof. dr. S. Hartmann

Overige leden van de Promotiecommissie:

prof. dr. J.A.G. Groenendijk
prof. dr. J.M. Sprenger
prof. dr. M. Werning
prof. dr. D. Zaefferer
dr. F.A.I. Buekens

For Sam,
for everything

et ego dico tibi quia tu es Petrus et super hanc petram ædificabo ecclesiam meam.

(*Vulgate*, Matthew 16: 18)

The type upon which the whole was constructed.

(**Hill and Topor, 1992**, p. 3)

Preface

Unification is one of the central aims of science. More than discovering large numbers of facts about the observable universe, scientists aim to establish these facts' common properties and relations. The strive for unification has several rationales: far from only promoting cognitive economy and simplicity, unification explains the success of one theory (or model) in terms of another, establishes their relative consistency, and effects a mutual flow of evidential support between the two theories.

This dissertation makes a contribution to the unificatory project of science. Its domain of unification constitutes the ontological 'zoo' of natural language semantics, cf. (**Bach, 1986**). This zoo comprises the plethora of objects which are assumed as the referents of certain classes of linguistic expressions. These include individuals (e.g. Bill), propositions (Bill walks), properties of individuals (walking), relations between individuals (finding), and many other kinds of objects.

The aim of this dissertation is to identify a single semantic basis for the above classes of objects, and to describe a bootstrapping procedure for them. Montague's formal semantics, cf. (**Montague, 1970a**; **1973**), makes a significant contribution to this goal: following Church's type theory, cf. (**Church, 1940**), Montague (**1970a**) reduces the referents of the small fragment of English from (**Montague, 1973**) to constructions out of *two* basic types of objects: individuals and propositions. From them, first-order properties of individuals and binary relations between individuals are constructed as functions from individuals to propositions, respectively as functions from ordered pairs of individuals to propositions. Yet, the question remains whether it is also possible to construct the ontological zoo from a *single*, rather than two, semantic bases. Recent research on language development (**Carstairs-McCarthy, 1999**; **2005**; **Cheney and Seyfarth, 1990**) points in this direction.

Partee (**2006**) takes first steps towards a *complete* unification of the linguistic ontology. To show the possibility of obtaining the ontological zoo from a *single* basic type of object, she sketches how a one-base-type system enables us to obtain several classes of linguistically relevant objects. However, the nature of her paper (a short *Festschrift* contribution) prevents the detailed presentation of this semantics. A proof of its workability is left to the semantic community.

This dissertation solves Partee's challenge. In particular, it provides a formal semantics for Montague's fragment of English which constructs the referents of all expressions of the fragment from a *single* basic type of object. The development of this semantics illustrates the strict requirements on a single-type semantics. The semantics based on Partee's preliminary single-type choice and other intuitively appealing alternatives fail to satisfy some of these requirements.

Acknowledgements

This project has its origins in a Fall 2007 visit to UMass Amherst. In preparation of my Master's thesis, I was there to discuss the foundations of formal semantics with Barbara Partee. For one of our meetings, Barbara brought a copy of her (**2006**) paper. To me, the main claim of the paper was initially mind-boggling: If the unification of Montague's basic types was not trivial (consider Frege's (**1891**) treatment of truth-values as individuals), it was sheer impossible (consider the empirical evidence for distinct name- and sentence-denotations). Partee's identification of a preliminary single-type candidate (**Partee, 2006**, p. 39) and her provision of a single-type model for a miniature fragment of English gave me some grasp of the intended unification. Yet, since I was still unable to envisage an extension of her model to larger fragments, I decided to put the paper aside for a while.

When I was applying for PhD positions in 'semantic Holland', I again stumbled across Partee's paper. Its claim now seemed to me even more pressing than before. Further, while I was still unsure about the identity of the single basic type, I could, at this time, develop a dedicated one-type logic and use it to model a Montague-style fragment of English. The logic and its application (which are included in Part I) were first presented at CiE 2010, cf. (**Liefke, 2013**), and at the *6th International Symposium of Cognition, Logic and Communication*. For the symposium, I had prepared some extra slides on the single-type domain (then, generalized quantifiers over individuals) and its interpretation of names and (in-)transitive verbs. However, since generalized quantifiers failed to encode relations between individual-representations, I soon discarded this basic-type choice.

I used the following year for a systematic identification of suitable single-type candidates. In this process, I was greatly helped by Reinhard Muskens, Daisuke Bekki, Zoltán Gendler Szabó, and Jim Pryor. In January 2011, Professor Bekki's invitation to Ochanomizu University enabled me to test my ideas on an audience of computer scientists. The ensuing discussion raised my awareness of the robustness of single-type semantics under the basic-type choice, and gave me a better grasp on the single-type representation of properties.

Over the last two-and-a-half years, this dissertation has gradually taken its shape. I owe much progress to my audiences at CiE 2010 and at the International

Symposium, at the *8th Formal Epistemology Workshop* (FEW 8), the ESF workshop *Philosophy of Computer Science and Artificial Intelligence*, the workshop *MCMP meets Linguistics*, GAP.8, the Chicago *Workshop in Semantics and Philosophy of Language*, Professor Bekki's research seminar in Information Science, the NII and Keio *Semantics Research Group*, the colloquium *Philosophy Meets Cognitive Science* (Bochum), the ASL *Logic Colloquium*, *Sinn und Bedeutung 18*, the *Investigating Semantics* conference, and the NYU *Semantics Group*. My colleagues from the TiLPS *Logic and Language* and *Epistemology and Philosophy of Science* seminars (Tilburg), the MCMP *Colloquium in Mathematical Philosophy*, the MCMP *Work in Progress Seminar*, and the LiPP Oberseminar *Diskussion Aktueller Linguistischer Arbeiten* (all LMU Munich) have given me valuable feedback.

The comments of my committee members Stephan Hartmann, Filip Buekens, Jeroen Groenendijk, Jan Sprenger, Markus Werning, and Dietmar Zaefferer have greatly improved an earlier version of this dissertation. In particular, Jeroen Groenendijk's comments on existence, aboutness, and related ideas have given this thesis a better conceptual and philosophical foundation. Groenendijk's question about the accommodation of individual concepts in single-type semantics has prompted me to extend the empirical scope of the presented semantics to the full fragment from (**Montague, 1973**). By suggesting a number of linguistic phenomena which can be modeled in single-type semantics, but which defy modeling in classical Montague semantics, Filip Buekens, Markus Werning, and Dietmar Zaefferer have helped provide single-type semantics with better linguistic illustrations and with a stronger empirical motivation.[1] Dietmar Zaefferer and Jan Sprenger have brought to my attention the analogy between the linguistic non-salience of a single basic type and the Sheffer stroke. Already at an earlier stage, Stephan Hartmann has raised my awareness of the epistemic advantages of using smaller basic-type sets.

Beyond the above, this work has profited from discussions with Chris Barker, Luca Barlassina, Daisuke Bekki, Alastair Butler, Lucas Champollion, Gennaro Chierchia, Daniel Cohnitz, Cleo Condoravdi, Dan Flickinger, Itamar Francez, Michael Glanzberg, Leon Horsten, Makoto Kanazawa, David Kaplan, Chris Kennedy, Jeremy Kuhn, Hannes Leitgeb, Sebastian Lutz, Salvador Mascarenhas, Reinhard Muskens, Thomas Müller, Eric Pacuit, Barbara Partee, John Perry, Stanley Peters, Chris Potts, Jim Pryor, Christian Retoré, Craige Roberts, Floris Roelofsen, Sam Sanders, Joseph Stern, Martin Stokhof, Zoltán Gendler Szabó, Anna Szabolcsi, Matt Teichman, Malte Willer, Greg Wheeler, Seth Yalcin, Shunsuke Yatabe, Ed Zalta, and Ede Zimmermann. Throughout this dissertation, I try to acknowledge their contribution in footnotes. Needless to say, none of these individuals are in any way responsible for the content or presentation of this work. Mistakes and confusions are entirely my own.

[1] More phenomena have been suggested by Dan Flickinger, Jeremy Kuhn, and Chris Potts.

I am very grateful to Stephan Hartmann for accepting me as a PhD student at Tilburg University, and for taking me to LMU Munich for the last one-and-a-half years of my dissertation. Stephan has been a wonderful supervisor and supporter, who has been a source of inspiration over the years.

Preparatory work for this dissertation has been made possible by two DAAD grants (grant D/04/43015 for studies at UCLA (2005–2006) and grant D/07/45595 for MA thesis-related research at UMass Amherst and Harvard (Fall 2007)). In the last few years, my participation in conferences has been made possible by Tilburg University (2009–2012), by the Association for Symbolic Logic (via student travel grants to the *Logic Colloquia*), by the Elsevier Foundation (via a women's travel grant to CiE 2010), by the European Science Foundation (via a travel grant for the ESF workshop), and by the University of Southern California (via a travel grant for FEW 8). Since my arrival in Munich (Fall 2012), I have received further support from LMU Munich, from the Alexander von Humboldt-Foundation (via Stephan Hartmann's Humboldt grant), and from LMU's women's mentoring program.

In the later phases of dissertation writing, I have spent three short-term visits at CSLI (Nov. 2012, 2013) and at the NYU Linguistics Department (Dec. 2013). I thank the LMU mentoring program for making these visits possible, and John Perry, Ed Zalta, and Anna Szabolcsi for making them fruitful and enjoyable. I further thank Andreas Weiermann and Kazuyuki Tanaka for their hospitality during my visits to my husband's research groups in Ghent and Sendai. From my time at CAU, I thank Claudia Claridge and Matthias Meyer for raising my interest in linguistics, George Pavlakos for encouraging my ventures into analytic philosophy, and Dirk Westerkamp for giving me more academic freedom than I deserved. From my time at UCLA, I thank Tony Martin and Yiannis Moschovakis for giving me my first logic classes, and David Kaplan for introducing me to Montague semantics and for introducing me to Barbara Partee.

On less academic matters, I thank my parents, Ingrid and Reinhard Liefke, for supporting my choice of an 'unusual' field of study, and – when the time came – for accepting my move abroad (first to Fort Worth, and then, via Los Angeles and Tilburg, to the other side of Germany). During my stay in the Low Countries, my parents-in-law, Jenny and Frans Sanders-Gotink, have taken me up as a member of their family. I thank them all for their love and support.

This dissertation is dedicated to my husband, Sam: Not only for standing by my side in good days (when the logic works) and in bad days (when it doesn't), but for volunteering as a guinea-pig for new ideas, for pushing me towards my goals, for putting up with my worldly oblivion, and – finally – for reminding me every now and so often that there is life beyond this dissertation.

Kristina Liefke
Munich, March 2014

Contents

Preface ix

Acknowledgements xi

List of Figures xix

List of Tables xxi

Version Information xxiii

Chapter 1. Introduction 1
1.1. Montague's Formal Semantics 1
1.2. Partee's Conjecture 8
1.3. Objective and Overview 17
1.4. Other Motivations for Partee's Conjecture 19
1.5. Intended Audience and Order of Reading 22
1.6. Sources of Chapters 24

Part I. 'Pure' Single-Type Semantics 27

Chapter 2. The 'Pure' Single-Type Logic TY_0 31
2.1. Types and Terms 31
2.2. Models 34
2.3. The Role of Metatheory 40
2.4. Entailment and Proof Theory 40
2.5. Ways to Truth 45
2.6. Summary 48

Chapter 3. TY_0-Based Single-Type Semantics 49
3.1. A Syntax for the PTQ Fragment 49
3.2. A TY_0 Semantics for the PTQ Fragment 53
3.3. PTQ Equivalence and Consequence 64
3.4. Explanatory Power of 'Pure' Single-Type Semantics 66
3.5. Summary 68

Part II. Single-Type Ontology 71

Chapter 4. Histoire d'*o* 75
4.1. Single-Type Requirements 75
4.2. Identifying Suitable Candidates 78
4.3. Single-Type Candidates and Partee's Conjecture 96
4.4. Summary 97

Chapter 5. Single-Type Methodology 99
5.1. Robustness in Single-Type Semantics 99
5.2. Classification of Single-Type Semantics 101
5.3. Levels of Single-Type Semantics 104
5.4. Summary 109

Part III. 'Mixed' Single-Type Semantics 111

Chapter 6. The Metatheory, TY_2^3 115
6.1. Types and Terms 115
6.2. Models 122
6.3. Truth 126
6.4. Entailment and Proof Theory 128
6.5. Summary 130

Chapter 7. Weak Single-Type Semantics 133
7.1. The 'Weak' Object Theory WTY_1^3 133
7.2. A WTY_1^3 Semantics for the PTQ Fragment 137
7.3. PTQ Truth and Equivalence 153
7.4. Sorting Single-Type Objects 157
7.5. Summary 160

Chapter 8. Strong Single-Type Semantics 163
8.1. The 'Strong' Object Theory STY_1^3 163
8.2. An STY_1^3 Semantics for the PTQ Fragment 166
8.3. PTQ Truth and Equivalence 179
8.4. A Note on Semantic Complexity 183
8.5. Summary 187

Chapter 9. Conclusion 189
9.1. Assessment of Single-Type Semantics 189
9.2. Constraints on Single-Type Semantics 192
9.3. Answering the Initial Questions 193
9.4. Precursors of Single-Type Semantics 194
9.5. Other Approaches to Single-Type Semantics 197

9.6. Future Work 199

Appendix A. Abbreviations and Conventions 203
A.1. List of Abbreviations 203
A.2. Notational Conventions 205
A.3. Glossary 207

Appendix B. Further Motivation for Partee's Conjecture 209
B.1. Partee's Original Motivation 209
B.2. New Motivation for Partee's Conjecture 213

Appendix C. Proofs 217
C.1. Derived Sequent Rules for TY_0 217
C.2. Proofs of Theorems 219
C.3. Definitions of Designated WTY_1^3 and STY_1^3 Constants 223

Appendix D. PTQ Translations and Definitions 227
D.1. Solving Partee's Temperature Puzzle 227
D.2. 'Weak' PTQ Definitions 233
D.3. 'Strong' PTQ Definitions 243

Bibliography 253

Summary 261

Samenvatting 263

Zusammenfassung 265

Abstract 267

Curriculum Vitæ 269

List of Figures

1.1 Indirect interpretation in the PTQ model. 3

1.2 Relations between type-e, -$\langle e, \langle s, t\rangle\rangle$, and -$\langle\langle e, \langle s, t\rangle\rangle, \langle s, t\rangle\rangle$ objects. 21

1.3 Relations between Fig. 1.2, type-$\langle s, t\rangle$, and single-type objects. 21

1.4 Alternative orders of reading. 24

2.1 An entity algebra. 38

2.2 A pentagon and a diamond. 39

3.1 The relation between DS, SS, LF, and PF. 50

3.2 Syntactic categories and TY_0 types. 68

4.1 Single-type candidates and their suitability. 78

4.2 An $\langle s, \langle s, t\rangle\rangle$-representation of John at @. 87

4.3 An $\langle s, \langle s, t\rangle\rangle$-representation of 'John loves Mary' at @. 89

4.4 An $\langle s, \langle s, t\rangle\rangle$-representation of John at partial @. 94

5.1 'Weak' and 'strong' representations of individuals/propositions. 102

5.2 Indirect interpretation in the 'pure' single-type model. 106

5.3 Defined doubly indirect interpretation. 107

5.4 Defined singly indirect interpretation. 107

6.1 The logical lattice $\mathbf{L_3}$. 124

7.1 Doubly indirect interpretation via WTY_1^3. 134

7.2 Syntactic categories and WTY_1^3 subtypes (1). 157

7.3 Syntactic categories and WTY_1^3 subtypes (2). 160

8.1 Syntactic categories and STY_1^3 subtypes. 183

9.1 Relations between Fig. 1.2, type-$(s;t)$, and type-$(s\,s;t)$ objects. 190

List of Tables

2.1 Structural rules for TY_0. 42
2.2 Logical rules for TY_0. 43
2.3 Some derived rules for TY_0. 44
2.4 Strategies for obtaining TY_0 truth. 46

3.1 Lexical insertion rules. 51
3.2 Phrase structure rules. 52
3.3 $\mathcal{L}$ constants. 55
3.4 TY_0 variables. 55

6.1 The Strong Kleene tables for $\wedge$, $\vee$, and $\neg$. 125
6.2 The Strong Kleene tables for $\rightarrow$ and $\Rightarrow$. 129
6.3 Additional logical rules for TY_2^3. 130

7.1 $\mathcal{L}^{w1}$ constants. 138
7.2 WTY_1^3 variables. 138
7.3 $\mathcal{L}^2$ constants. 139
7.4 TY_2^3 variables. 139
7.5 Sorted $\mathcal{L}^{w1}$ constants. 159
7.6 Sorted WTY_1^3 variables. 159

8.1 $\mathcal{L}^{s1}$ constants. 166
8.2 STY_1^3 variables. 167

9.1 Success of single-typing vs. Montague typing. 192
9.2 Conditions on the success of single-type semantics. 193

Version Information

Current version: 2.0

Version 1.0:	refereed manuscript	(June 2013)
Version 1.1:	revised manuscript	(January 2014)
Version 2.0:	printed version	(March 2014)

CHAPTER 1

Introduction

This dissertation is a contribution to the *formal semantics* of *natural language*. *Natural languages* are human languages like English or Japanese, which are used by ordinary people in everyday communication. *Formal semantics* is an approach to the study of natural language meaning which interprets linguistic expressions through the use of mathematical and logical models. In virtue of this interpretation, formal semantics explains our ability to derive the meaning of complex expressions (e.g. the meaning of the sentence John loves Mary) from the meanings of their syntactic constituents (here, John, love, and Mary), formulates truth- and equivalence-conditions for sentences, and characterizes their relation of entailment.

The present chapter provides the linguistic background to this dissertation[1], and introduces its topic and objective: Section 1.1 presents the main ingredients of formal semantics, indicates their philosophical interest, and sketches a number of recent developments. Section 1.2 presents a reaction to one such development: the reduction of prolifering semantic types to a *single* basic type. Sections 1.2.1, 1.2.2, and 1.4 review different motivations for the proposed 'single-type' semantics. Section 1.3 specifies the aim of this dissertation, and gives an overview over its contents. Section 1.5 suggests orders of reading for different audiences.

1.1. Montague's Formal Semantics

Formal semantics has its origins in the work of Frege, Carnap and Tarski, and was developed in the early 1970's by David Lewis, Donald Davidson, and Max Cresswell. However, its single most important influence is the work of Richard Montague, cf. (**Montague, 1970a**; **1970b**; **1973**).[2] Montague (1930–1971) was a California-trained logician and philosopher, who held a pronounced interest in the mathematical treatment of natural language. In particular, Montague believed that natural languages could be described as *interpreted formal systems*. This view, called *Montague's thesis* (**Bach, 1986**), is expressed in (**Montague, 1970b**, p. 222):

> There is in my opinion no important theoretical difference between natural languages and the artificial languages of logici-

[1] Readers who are familiar with formal semantics may proceed directly to Section 1.2.

[2] As a result, we will sometimes refer to formal semantics as *Montague semantics*.

> ans; indeed I consider it possible to comprehend the syntax and semantics of both kinds of languages within a single natural and mathematically precise theory.

Montague's thesis was in stark contrast to the received view of formal and natural languages at the time: While most generative linguists doubted the appropriateness of logical approaches to semantics, most logicians believed that natural languages resisted a precise formalization. This belief was motivated by the restriction of traditional logical tools to the apparatus of first-order predicate logic, and the existence of a *mismatch* between the grammatical form of disambiguated natural language sentences and the logical form of their predicate-logical translations.[3] To refute this belief, Montague replaced predicate logic by a variant (called *Intensional Logic*, IL) of the lambda logic from (**Church, 1940**), cf. (**Henkin, 1950**). Since the language of IL allows abstraction over objects of any order, its terms can take a very similar form to the grammatical form of natural language.

1.1.1. Central Aspects. To introduce the reader to Montague's formal semantics[4], we present some of its central aspects. These include the use of model-theoretic semantics, the adoption of the method of indirect interpretation, the centrality of the principle of compositionality, and the particular role of type theory.

Model-Theoretic Semantics. Model-theoretic semantics, cf. (**Tarski, 1933**), is an approach to the semantics of natural language which interprets linguistic expressions as elements (objects, sets, or functions) in the domains of mathematical models. These elements provide the meaning (or *semantic value*) for every expression of a given subset of natural language. Expressions are assigned a value by interpretation functions. In particular, these functions send proper names to basic elements (i.e. *individuals*), and send intransitive verbs to *sets* of individuals.

To model intensional and modal phenomena, Montague (**1973**) enriches his models with a domain of possible world-time pairs (or *indices*). This enrichment enables the interpretation of linguistic expressions as functions from indices to the expressions' values at those indices. Thus, declarative sentences are interpreted as *sets* of indices at which they are true. Intransitive verbs are interpreted as *functions* from indices to sets of individuals which witness the denoted property.

As a result of Montague's interpretation of sentences, a sentence S is true at a given index w w.r.t. some model iff w is a member of the interpretation of S in the model. Entailment then becomes definable as truth-preservation: The sentence S entails a sentence S' iff S' is true at all indices at which S is true.

[3]Thus, the predicate-logical translation of the sentence John finds a unicorn, i.e. $\exists x.\, unicorn\,(x) \wedge find\,(\underline{x}, john)$, scatters the translation of the NP a unicorn (underlined) over the whole formula.
[4]Good introductions to formal semantics include (**Gamut, 1991**), (**Dowty et al., 1981**), and (**Heim and Kratzer, 1998**).

Indirect Interpretation. To make the model-theoretic interpretation of natural language more perspicuous, Montague (**1973**, **1970b**) uses an *indirect* interpretation of natural language, which proceeds via the translation of a subset (or *fragment*) of natural language into the language of some logic (here, the language of IL). The semantic interpretation of natural language thus constitutes a three-step process, which involves the syntactic formalization of a non-trivial fragment of natural language (here, the fragment from (**Montague, 1973**)) (step 1), the development of a language ($\mathcal{L}$), domain ($\mathcal{F}$), and interpretation function ($\mathcal{I}$) for the interpreting logic IL (step 2), and the provision of a set of translation rules sending linguistic expressions (or *logical forms*) of the fragment to IL terms (step 3).

Figure 1.1 illustrates the obtaining of the interpretation, $\mathcal{I}(\chi)$, of a linguistic expression X via its translation into the logical term χ. In the figure, 'LF' designates the Logical Form-component of Montague's fragment.

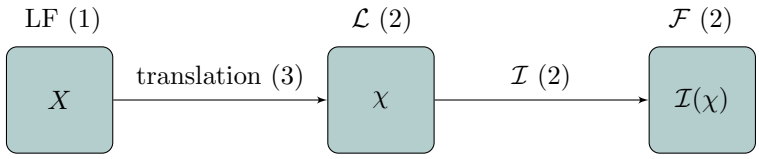

FIGURE 1.1. Indirect interpretation in the PTQ model.

Compositionality. To ensure the success of the method of indirect interpretation, the translation of natural language into the language of IL must adhere to the *principle of compositionality* of translations. This principle is a syntactic version of the principle of (semantic) compositionality, cf. (**Partee, 1984**; **Janssen, 1986**; **Hodges, 2001**), which requires the existence of an IL translation for every syntactically basic expression (or *word*), and the existence of an IL correspondent (here, functional application) for every syntactic operation (e.g. merging):

> **Principle of Compositionality:** *The meaning of an expression is a function of the meanings of its constituents and their mode of combination.*

In Montague (**1973**), cf. (**Montague, 1970b**), the compositional translation of linguistic expressions is made possible by the identification of IL with a variant of Church's lambda logic, such that there exists a translation for every linguistically basic word (including determiners (the, a/some) and quantifiers (every)).

The compositional *interpretation* of linguistic expressions is enabled by the interpretation of IL terms as functions from indices to *sets* of elements in models. As a result, the meanings of basic non-sentential expressions are identified with their contribution to the truth-conditions of the sentences in which they occur.

The meanings of intransitive verbs (e.g. walks) fulfill this function by sending indices to *sets* of individuals. Thus, the application of an index-specific interpretation of walk (in a given model) to the interpretation of Bill yields the truth-value of the sentence Bill walks at the index.

The possibility of obtaining the truth-conditions of *molecular* sentences (e.g. the sentence Bill walks *and* John whistles) at an index is enabled by the identification of IL models with *set-theoretic* models. As a result, the semantic correspondents ($\cap$, $\cup$, $-$) of the familiar logical constants (e.g. $\wedge$, $\vee$, resp. $\neg$) are readily available in IL. For example, since linguistic conjunction is associated with set intersection, the sentence Bill walks and John whistles is interpreted as the intersection of the sets of indices which interpret the sentences Bill walks and John whistles.

Type Theory. To capture the relations between the interpretations of expressions from different syntactic categories, Montague casts the structure on a model's domains into a *type system*, cf. (**Russell and Whitehead, 1997**; **Curry, 1934**; **Church, 1940**).[5] A type system is a pair of domains and domain constructors, which enables the formation of other, more complex domains (e.g. function spaces). In particular, the type system of the logic IL assumes two basic types of objects: individuals (type e) and truth-values (type t). From them, complex types are formed via the rules **CT** and **IT** (for *Church types*, resp. for *intensional types*), cf. (**Montague, 1973**). Indices (type s) are introduced through the second rule:

(**CT**) *If α and β are types, then $\langle \alpha, \beta \rangle$ is the type for functions from objects of the type α to objects of the type β.*

(**IT**) *If α is a type, then $\langle s, \alpha \rangle$ is the type for functions from indices (type s) to objects of the type α.*

The first rule identifies the type $\langle e, t \rangle$ as the type for functions from individuals to truth-values (or, equivalently, for sets of individuals). The second rule identifies the type $\langle s, \langle e, t \rangle \rangle$ as the type for functions from indices to sets of individuals. This type is associated with the semantic values of intransitive verbs.

The typing of linguistic expressions provides a formal basis for syntactic categories. As a result, we can use typing to check the well-formedness of linguistic expressions, and to explain a large number of distributional phenomena. For example, since the words John, Mary, and run receive an interpretation in the types e (for John, Mary) and $\langle s, \langle e, t \rangle \rangle$ (for run), the string John runs Mary does not qualify as a well-formed structure of English.[6] Finally, the system only allows predicative constructions, thereby avoiding the usual Russell-style inconsistencies.

[5](**Muskens, 2011**) and (**Turner, 1997**) are excellent expositions of type theory. The introduction to (**Barendregt et al., 2010**) contains a motivation of its historical introduction.

[6]After the interpretation of runs at an index has been applied to the interpretation of Mary, there is no longer a free type-e argument place for the interpretation of John.

This completes our presentation of the main ingredients of Montague's formal semantics. We next discuss their philosophical interest.

1.1.2. Philosophical Interest. Montague semantics has provided the topics for many philosophical discussions. This is due, in part, to the theory's philosophical origins[7], and to Montague's close interaction with leading philosophers of the time.[8] However, the last 30 years have seen an active philosophical discussion of many of Montague's *technical* choices (i.e. of the *foundations* of formal semantics), cf. (**Fox and Lappin, 2005**). These choices include Montague's adoption of the principle of compositionality, his characterization of linguistic meaning, and the ontology of his type theory. We discuss these three topics in turn:

Compositionality. Montague's adherence to the principle of compositionality is a direct consequence of his mathematical view on natural language, cf. (**Montague, 1970b**, p. 222): If natural languages are describable as interpreted formal systems, we expect that they also share the formal properties of these systems (specifically, the existence of a structure-preserving map between the algebra of the logical language and the algebra of its semantic interpretations).

Compositionality is consistent with the assumed productivity and systematicity of natural languages. However, the last 35 years have produced many putative counterexamples to compositionality, cf. (**Janssen, 1997**; **Szabó, 2012**). These counterexamples include instances of cross-sentential anaphora, in which the meaning of the relevant complex expression also depends on the expression's linguistic *context*. The identification of productive (or systematic) languages which resist a compositional treatment has recently questioned the above-cited reasons for compositionality, cf. (**Werning, 2005**). Philosophical interest in compositionality regards the justification of compositionality, its characterization as a theoretical or as a methodological principle, the role of context in the interpretation of linguistic expressions, and the reconciliation of compositionality with its counterexamples.

Meaning and Reference. Montague (**1973**) provides a model-theoretic treatment of natural language meanings which characterizes meanings as *intensions*, cf. (**Carnap, 1988**). The intension of an expression is a function from indices to the expression's semantic values at those indices (i.e. to its *extensions*). An expression's extension is obtained by evaluating its intension at the current index.

Montague's *modal* treatment of intensionality (via the introduction of *indices* into type-theoretic models; cf. Sect. 1.1.1) gives rise to a number of philosophical questions: What is the metaphysical status of indices? Are individuals identical across indices, or do they have Lewisian (other-index) counterparts? How is the

[7]Some origins of Montague semantics are Tarski's (**1944**) theory of truth, Carnap's (**1988**) theory of intensions and extensions, and Kripke's (**1959**) possible world semantics.
[8]The latter include Benson Mates, David Lewis, Terence Parsons, and David Kaplan.

counterpart relation defined? Other questions regard the empirical adequacy of intensions: Are intensions sufficiently fine-grained to capture speakers' intuitions about strict synonymy? Do they enable correct predictions about linguistic entailment? Do they allow the suitable modeling of propositional attitude statements?

Type Theory and Ontology. Natural languages presuppose a rich semantic ontology: To interpret the fragment of English from (**Montague, 1973**) (here, the *PTQ fragment*), we require the existence of individuals (e.g. Bill), propositions ('Bill walks'), first- and higher-order properties of individuals ('walks', 'is one of Bill's properties'), binary relations between individuals ('find'), and many other kinds of objects. Montague (**1970a**) reduces this large set of primitives to constructions (via the rule **CT**) out of *two* basic types of objects: individuals (type e, for '*entities*') and propositions (or functions from indices to truth-values, type $\langle s,t\rangle$).[9] First- (or second-)order properties of individuals are then represented as (functions from) functions from individuals to propositions (to propositions). Binary relations between individuals are represented as functions from individuals to functions from individuals to propositions, etc.

The reduction of semantic primitives to individuals and propositions unifies the semantic ontology of the PTQ fragment, and establishes new representational relations between objects of different types.[10] Philosophical interest in Montague's type theory further concerns the identity of the basic types, their interchangeability with other basic types (which also construct all classes of PTQ referents), and semantic requirements on these types (e.g. the existence of an algebra on one type's domain). The last three topics will be discussed in this dissertation.

1.1.3. Recent Developments. The past 35 years have seen a number of revisions and extensions to Montague semantics. The former include the streamlining of Montague's Intensional Logic to a logic with more desirable proof-theoretic properties, cf. (**Gallin, 1975**), and with a simpler (or more easily generalizable) model theory, cf. (**Muskens, 1989**). The latter include the improvement of the empirical adequacy of Montague semantics, cf. (**Thomason, 1980**), its application to other languages like German, cf. (**Löbner, 1976**), or Japanese, cf. (**Bekki, 2010**), and its extension to larger fragments of natural language containing, e.g., plurals, mass, and kind terms (**Link, 1983**; **Chierchia, 1998**), neutral perception verbs (**Muskens, 1989**), impredicative constructions (**Chierchia and Tur-**

[9]Our adoption of the basic types e and $\langle s,t\rangle$ (rather than of the types e and t, cf. (**Montague, 1973**), e, s, and t, cf. (**Gallin, 1975**), or $\langle s,e\rangle$ and $\langle s,t\rangle$, cf. (**van Eijck and Unger, 2010**; **Fox et al., 2002**)) is motivated by our wish to parallel the syntactic distinction between proper names and sentences, and by Partee's choice of the type e as a basic type, cf. (**Partee, 2006**, p. 37). We will show in Chapter 3 and Appendix D.1 that a semantics with the basic types e and $\langle s,t\rangle$ can still model intensional phenomena (incl. a solution to Partee's temperature puzzle).

[10]For example, the representation of first-order properties suggests the possibility of representing individuals via functions from individuals to the set of indices at which the individuals exist.

ner, 1988), adverbial modifiers (**Dowty, 1979**), cf. (**Davidson, 1967**), scalar adjectives (**Cresswell, 1976**), and anaphora (**Muskens, 1996**; **Bekki, 2012**).

These adaptations all involve some deviation from Montague's original type system. In particular, the semantics from (**Thomason, 1980**), (**Muskens, 1989**; **1996**), (**Dowty, 1979**), (**Cresswell, 1976**), and (**Chierchia and Turner, 1988**) enrich the IL type system with types for primitive *propositions*, *situations*, *registers*, *events*, *states*, *processes*, *intervals*, *degrees*, *numbers*, and *kinds*. The semantics from (**Fox and Lappin, 2005**) and (**Bekki, 2012**) further supplement the type-forming rule **CT** with rules for the formation of separation, comprehension, and polymorphic types, cf. (**Curry, 1934**; **Girard, 1972**), and of dependent types, cf. (**Martin-Löf, 1973**).

Since some of the above-proposed types (e.g. Muskens' type for situations) constitute generalizations of other types (possible worlds, or indices), the presented extensions to the Montagovian type system need not all be simultaneously implemented.[11] Yet, the accommodation of the above phenomena in a single type system still induces the adoption of around *ten* (instead of *two* or *three*) basic types.

The extension of the Montagovian type system is a consequence of the institutionalization of contemporary formal semantics as a branch of *linguistics*, and the attendant emphasis on practical applications of this system. The availability of a larger number of ontological primitives facilitates work for the empirical linguist: In a rich type system, fewer syntactic expressions are interpreted in a complex type.[12] As a result, the compositional translations of many syntactic structures will be *simpler*, and will involve *less* lambda conversions than their IL counterparts.

But the proliferation of basic types is not an altogether positive development. Specifically, by centering their attention on the simplicity of application, many contemporary formal semanticists have lost sight of Montague's original methodological objective (i.e. the treatment of natural language as a *simple* and *elegant* mathematical theory). In particular, the replacement of Montague's type system by systems with *more* basic types *reduces* the number of representational relations between different types of objects[13], and *decreases* the resulting unificatory effect on the semantic ontology. (Situation Semantics, cf. (**Barwise and Perry, 1983**; **Kratzer, 1989**), and Property Theory, cf. (**Chierchia et al., 1988a**; **1988b**), are pleasant exceptions to this development).

---

[11] This is, in particular, due to the characterization of situations as *parts* of possible worlds (or as *partial possible worlds*), cf. (**Barwise and Perry, 1983**).

[12] For example, since linguists typically assign degree modifiers (e.g. very) the type for degrees d (rather than the type for second-order properties $\langle\langle e, \langle s, t\rangle\rangle, \langle s, t\rangle\rangle$), gradable adjectives (e.g. tall) receive a translation in the type $\langle d, \langle e, \langle s, t\rangle\rangle\rangle$, instead of the type $\langle\langle\langle e, \langle s, t\rangle\rangle, \langle s, t\rangle\rangle, \langle e, \langle s, t\rangle\rangle\rangle$.

[13] Thus, the treatment of degree modifiers as type-d expressions prevents the identification of their relation to individuals (type e) and individual-properties (type $\langle e, \langle s, t\rangle\rangle$; $\langle\langle e, \langle s, t\rangle\rangle, \langle s, t\rangle\rangle$).

This dissertation inverses the observed explosion of basic types in formal semantics: Instead of extending the IL type system via the *introduction* of *new* basic types, it attempts to *reduce* its members to a *single basic type*. This project follows the impetus of Barbara Partee, cf. (**Partee, 2006**). To constrain its scope, we restrict ourselves to Montague's PTQ fragment (which does not contain mass terms, neutral perception verbs, impredicative constructions, scalar adjectives, or cross-sentential anaphora), and neglect arguments for the existence of events, cf. (**Davidson, 1967**; **Parsons, 1990**).

At this point, the specialist reader will enthusiastically interject proposals of the form *Wouldn't X be a good single-type candidate?* (where X is a familiar type from the formal semantic analysis of some linguistic phenomenon). In anticipation of our later results, we answer with Proposition 1.1:

PROPOSITION 1.1 (HYH). *The salient candidates prove unsuitable as a single basic type.*

The above fact will be established in Part II, Chapter 4.

1.2. Partee's Conjecture

This dissertation is an experiment: What happens if we replace Montague's types for individuals (e) and propositions ($\langle s,t\rangle$) by a single basic type of object? Is this possible? And, if yes, under what conditions? What does a suitable interpretive domain for the single basic type look like? What are its properties? What effects does this change of type system have on our semantics' ability to model natural language? How does it influence our understanding of the relations between different types of objects? Does it make Montague's type system dispensable?

The assumption behind the above questions, i.e. that the PTQ fragment has an even simpler semantic basis than the one adopted in (**Montague, 1970a**), has first been proposed by Barbara Partee. In particular, (**Partee, 2006**) makes the following suggestion about the linguistic type system:

PROPOSITION 1.2 (Single-Type Hypothesis). *The distinction between individuals and propositions is inessential for the construction of a rich linguistic ontology. The PTQ fragment can be modeled through the use of <u>one</u> basic type of object.*

To acknowledge its original proponent, we will sometimes call Proposition 1.2 *Partee's conjecture*. This conjecture suggests the possibility of obtaining all classes of PTQ referents from a single basic type (dubbed 'o')[14], whose objects capture the semantic content of individuals and propositions. From them, objects of a complex type are constructed via a variant, **ST** (for *<u>single-type</u>* rule), of the rule **CT**:

(**ST**) *If α and β are <u>single-types</u> types, then $\langle\alpha,\beta\rangle$ is a <u>single-type</u> type.*

[14]The basic type label 'o' has been suggested by Floris Roelofsen.

In virtue of the neutrality of the type o between Montague's types e and $\langle s,t\rangle$, any semantics which satisfies Proposition 1.2 (hereafter, *single-type semantics*[15]) will identify basic-type objects with the values of proper names (e.g. Bill; traditionally, type e) and of sentences and complement phrases (e.g. Bill walks, resp. that Bill walks; traditionally, type $\langle s,t\rangle$). As a result, it will also assign the same type, $\langle o,o\rangle$, to common nouns (e.g. man; traditionally, type $\langle e,\langle s,t\rangle\rangle$) and to complementizers and sentence adverbs (that, resp. possibly; traditionally, type $\langle\langle s,t\rangle,\langle s,t\rangle\rangle$). The types of expressions from all other syntactic categories of the PTQ fragment are obtained by replacing the labels 'e' and '$\langle s,t\rangle$' by the label 'o' in their associated Montague type.

Partee's conjecture about the possibility of a single-type semantics suggests a 'minimality test' for the Montagovian type system: If we can formulate a single-type semantics *without* reference to Montagovian individuals or propositions, we will therewith *refute* the commonly assumed need for two distinct basic types. If our formulation of a single-type semantics *relies* on the availability of Montagovian individuals or propositions, the semantics will *support* Montague's basic-type distinction.

However, our interest in single-type semantics is also motivated by many other considerations: These include *empirical* considerations (which regard the modeling power of single-type semantics w.r.t. traditional Montague semantics; cf. (**Partee, 2006**, Ingredients 4–5, 7)), *formal* considerations (which regard the possibility of constructing single-type models; cf. (*ibid.*, Ingredients 1–3, 6)), and other *methodological* considerations besides minimality testing. To illustrate possible applications of a single-type semantics – and to prime the reader's intuitions about such a semantics –, empirical and formal considerations are discussed in the remainder of this section. Methodological considerations, which drive our interest in single-type semantics, will be the subject of Section 1.4.

1.2.1. Empirical Considerations. Empirical motivations for Partee's conjecture lie in the observation that single-type semantics improves upon the modeling power of traditional Montague semantics. This improvement is a consequence of the neutralization of the distinction between the semantic types for proper names and sentences, such that there are *fewer* same-level constraints on semantic merging.[16, 17]

[15]Since such semantics still assume a type hierarchy over the basic type o, they should more correctly be referred to as '*single-base-type semantics*'. I owe this observation to Jim Pryor.

[16]As a result, transitive verbs (e.g. remember; traditionally, type $\langle e,\langle e,\langle s,t\rangle\rangle\rangle$; now, type $\langle o,\langle o,o\rangle\rangle$) can apply either to a proper name or to a complement phrase (now, both type o).

[17]Initially, the neutralization of the distinction between the types e and $\langle s,t\rangle$ *reduces* the explanatory power of single-type semantics (s.t. this semantics will be unable to explain distributional phenomena involving proper names, CPs, and sentences). We will restore the explanatory power of single-type semantics over the course of this dissertation.

To illustrate the higher modeling power of single-type semantics, we identify a number of linguistic phenomena which can be accommodated[18] in a single-type semantics, but which defy accommodation in the semantics from (**Montague, 1973**; **1970a**) (hereafter, *traditional Montague semantics*). Such phenomena occur in lexical syntax, the syntax of coordination, the semantics of specification, and non-sentential speech. They include the neutrality of certain classes of expressions between an NP or a CP complement, cf. (**Kim and Sag, 2005**; **Sag et al., 2003**), the possibility of coordinating NPs with complement phrases, cf. (**Bayer, 1996**; **Sag et al., 1985**), the existence of specificational sentences with a postcopular CP, cf. (**Potts, 2002**), and the use of proper names to assert a contextually salient proposition about their type-e referent, cf. (**Carstairs-McCarthy, 2005**; **Merchant, 2008**).

Below, we discuss these phenomena in turn. Since Partee's original empirical motivation for Proposition 1.2, i.e. the evolutionary contingency of the distinction between noun phrases and sentences, cf. (**Carstairs-McCarthy, 1999**), only weakly supports the possibility of a single-type semantics (via the assumption of a close relation between syntactic categories and semantic types, cf. (**Montague, 1970b**)), its presentation is deferred to Appendix B.1.

We start by showing how single-type semantics accommodates the phenomenon from lexical syntax.

Lexical Syntax. In (**Kim and Sag, 2005**), cf. (**Sag et al., 2003**; **Kim, 2008**), Kim and Sag observe that many verbs select a complement which can be realized as a noun phrase or a complement phrase. Thus, in (1), the verb remember can combine either with the name Bill (in (1a)) or with the CP that Bill was waiting for her (in (1b)). A similar observation can be made for the verbs fear and notice (in (2), (3)), and for many other factive, cognition, and experiencer verbs.

(1) a. Pat remembered [$_{\text{NP}}$Bill].
b. Pat remembered [$_{\text{CP}}$that Bill was waiting for her].

(2) a. Sherlock fears [$_{\text{NP}}$Moriarty].
b. Sherlock fears [$_{\text{CP}}$that Moriarty will destroy him].

(3) a. The librarian noticed [$_{\text{NP}}$*Moby Dick*].
b. The librarian noticed [$_{\text{CP}}$that *Moby Dick* was displayed in the window].

Since traditional Montague semantics assumes a *functional* relation between syntactic categories and semantic types (s.t. each category is associated with *exactly*

[18]Since there are other, more conservative, ways of accommodating these phenomena (cf. pp. 14–15), these phenomena do not constitute *evidence* for Partee's conjecture. As a result, we only understand empirical motivations for single-type semantics as *illustrations* of the empirical applications of this semantics.

one type), it cannot associate the different occurrences of the verbs from (1) to (3) with distinct types (e.g. with the types $\langle \underline{e}, \langle e, \langle s, t\rangle\rangle\rangle$ and $\langle \underline{\langle s, t\rangle}, \langle e, \langle s, t\rangle\rangle\rangle$)[19]. However, Montague's different type-assignment to proper names (type e) and CPs (type $\langle s, t\rangle$) would require such an association. As a result, traditional Montague semantics is unable to model at least one of the members of the above sentence pairs. For example, by assigning the verb remember the type $\langle e, \langle e, \langle s, t\rangle\rangle\rangle$, Montague semantics would preclude the interpretation of sentences of the form of (1b).[20]

The 'disabling' features of Montague semantics for the modeling of the pairs of sentences from (1) to (3) are summarized below:

OBSERVATION 1.1. *In Montague semantics, cf.* (**Montague, 1970a**)*, both of the following hold:*

(i) *Proper names receive an interpretation in the domain of the type e. Sentences and complement phrases receive an interpretation in the domain of the type $\langle s, t\rangle$.*

(ii) *No two occurrences of an expression receive an interpretation in the domains of different types.*

Single-type semantics solves the problem of accommodating NP/CP complement-neutral verbs by dropping the assumption of different type-assignments from Observation 1.1.i. In particular, by replacing the types e and $\langle s, t\rangle$ by the basic type o, this semantics enables the same-type interpretation of proper names and complement phrases. Since the new type of transitive verbs, $\langle o, \langle o, o\rangle\rangle$, will thus allow its expressions to take a name *or a CP* as its complement, it enables the interpretation of both members of the sentence-pairs from (1) to (3).

Syntactic Coordination. The same-type interpretation of proper names and complement phrases further enables single-type semantics to accommodate coordinate structures with a proper name- and a CP conjunct. Such structures include the results (in (4)–(6)) of coordinating the complements of the occurrences of the verbs from (1) to (3).[21] In the literature on coordination, these structures are described as *coordinations of unlike categories*, cf. (**Sag et al., 1985**; **Bayer, 1996**).

(4) Pat remembered [$_{\text{NP}}$Bill] *and* [$_{\text{CP}}$that he was waiting for her].

(5) Sherlock fears [$_{\text{NP}}$Moriarty] *and* [$_{\text{CP}}$that Moriarty will destroy him].

[19]For perspicuity, the type of the complement is underlined.

[20]One could attempt to obtain the required modeling power by introducing a different lexical entry for each occurrence of the verbs from (1) to (3), by assigning the different entries the types $\langle e, \langle e, \langle s, t\rangle\rangle\rangle$, resp. $\langle \langle s, t\rangle, \langle e, \langle s, t\rangle\rangle\rangle$, and by relating their semantic values through the use of suitable postulates. Yet, since this strategy is a hard-coded variant of the strategy from flexible Montague grammar (cf. pp. 14–16) – and since this differentiation of entries is not reflected in lexicographic research (cf. the **OED** entry for remember) –, we here ignore this strategy.

[21]For better visibility, we hereafter italicize coordinating conjunctions.

(6) The librarian noticed [$_{\text{NP}}$*Moby Dick*] *and* [$_{\text{CP}}$that it was displayed in the window].

In particular, the unified type for proper names and complement phrases, o, allows the coordination of expressions of unlike *basic* categories (i.e. NP and CP) under the satisfaction of the coordinability requirement from (**Montague, 1973**), cf. (**Partee and Rooth, 1983**). This requirement is stated below:

COORDINABILITY REQUIREMENT. *To allow coordination, linguistic expressions must receive an interpretation in the domain of the same semantic type.*

In traditional Montague semantics, the coordination of expressions of unlike categories is disabled by the interpretation of proper names and complement phrases in the domains of different types (cf. Obs. 1.1.i).

Specification. The advantages of single-type semantics over traditional Montague semantics are further illustrated by the ability of single-type semantics to model CP equatives.[22] The latter are copular sentences of the form of (7), cf. (**Potts, 2002**, pp. 67–68), which equate the referents of the two expressions flanking the copula. In contrast to typical equatives (whose arguments are both noun phrases; cf. (8)), CP equatives take as arguments an NP and a *complement phrase*.

(7) a. [$_{\text{NP}}$The problem] is [$_{\underline{\text{CP}}}$that Mary hates Bill].
b. [$_{\text{NP}}$The discovery] was [$_{\underline{\text{CP}}}$that there exists an eighth planet].

(8) [$_{\text{NP}_1}$The best singer] is [$_{\text{NP}_2}$Joan].

The assumption of an NP *and a CP* argument poses a challenge for the interpretation of CP equatives in traditional Montague semantics. This is due to the fact that the familiar interpretation of the copula – which demands that the copula's arguments have the same type (i.e. e), cf. (**Heycock and Kroch, 1999**) – does not allow its application to a proposition. But this is required for the modeling of the two sentences from (7). The introduction of an alternative interpretation of the copula (s.t. it allows pairs of type-$\langle s,t\rangle$ and type-e arguments) or of a nominalization function on propositions[23] (s.t. the CP argument is also interpreted in the type e) is prevented by Observation 1.1.ii.

The interpretation of referential noun and complement phrases in the single basic type o enables the interpretation of the two sentences from (7).

The merits of single-type semantics in the modeling of phenomena from lexical syntax, specification, and coordination are complemented by the ability of this semantics to model genuinely *semantic* phenomena. These phenomena include the interpretation of isolated occurrences of proper names in the semantic type for

[22] I owe this observation to Chris Potts.
[23] This function, cf. (**Potts, 2002**, p. 69, cf. pp. 57–58), sends propositions (type $\langle s,t\rangle$) to their individual correlates (type e).

sentences, and the identification of the semantic values of names in a given context with propositions which are denoted by salient sentences in this context.

Nonsentential Speech. Recent research in nonsentential speech, cf. (**Carstairs-McCarthy, 2005**; **Merchant, 2008**), has found that syntactically isolated occurrences of proper names in a given context can be interpreted as the result of applying a contextually salient property to the name's type-e referent. Thus, the name Barbara Partee – when uttered as a woman is entering the room – is interpreted as the sentence from (9b) (or (9c)) (**Merchant, 2008**, pp. 9, 25–26), cf. (**Stainton, 2006**, p. 6), rather than as the individual Barbara Partee:

(9) CONTEXT: A woman is entering the room. A linguist turns to her friend, gestures towards the door, and says (a).
   a. [$_{\text{NP}}$Barbara Partee]
   b. [$_{\text{NP}}$Barbara Partee] is (the woman) entering the room.
   c. [$_{\text{NP}}$Barbara Partee] is arriving.

Similarly, the expression Rob's mom – when uttered as Mia is lamenting strawberry chunks in her jam – allows an interpretation as the sentence from (10b) (**Merchant, 2008**, pp. 9, 25), cf. (**Stainton, 2006**, p. 113):

(10) CONTEXT: Mia is lamenting the strawberry chunks in her jam. Her mother nods understandingly and says (a).
   a. [$_{\text{NP}}$Rob's mom]
   b. [$_{\text{NP}}$Rob's mom] is responsible for the strawberry chunks in Mia's jam.

Since traditional Montague semantics does not interpret proper names in the semantic type for sentences (cf. Obs. 1.1.i), it is unable to model the phenomena from (9) and (10). Single-type semantics, which assigns the type o to both names and sentences, enables the accommodation of these phenomena.

But the empirical scope of single-type semantics is not restricted to the sentence-type interpretation of names. The semantics further accommodates the propositional *behavior* of names, which cannot be modeled in Montague semantics.

The same-type interpretation of proper names and sentences in single-type semantics suggests that names display the semantic behavior of sentences in this semantics: If names receive an interpretation in the same domain as sentences, we expect that names – like sentences – can be evaluated as true or false with respect to a given set of contextual parameters, and that they may be related[24] by semantic equivalence. This is indeed the case. In particular, in the situation from (9), the announcement (9a) – when the new arrival is, in fact, Angelika Kratzer – is a false statement, rather than a mere misidentification (**Stainton, 2006**, pp. 8–10,

[24]to other names, or to sentences.

16), cf. (**Carstairs-McCarthy, 2005**, p. 151). Similarly, if, in (10), the chunks of strawberries in Mia's jam cannot be traced back to Rob's Mom (be it that Rob's Mom did not make the jam, or that Rob sneaked in the strawberry chunks when she was not watching), the utterance of (10a) in that situation is simply false.

In virtue of their truth- and falsity-conditions, names of the above form will, in a given situation, be equivalent to all true sentences in this situation which carry information about the names' type-*e* referent. For example, if the new arrival in the above-described situation is indeed Barbara Partee, the utterance of the name from (9a) will be equivalent to the sentence from (9b) (or (9c)) in that situation. Similarly, if the cause of the strawberry chunks in Mia's jam is indeed Rob's mom, the utterance of (10a) will be equivalent to the sentence from (10b) in the described situation.

The obtaining of semantic equivalence relations between sententially interpreted noun phrases and sentences (or CPs) is supported by the assertion of an equivalence relation between the noun and complement phrases in the two sentences from (7). The obtaining of this relation ensures that the replacement of an NP (or CP) by its CP- (or NP-)equivalent in the complement of an NP/CP complement-neutral verb does not change the truth-value of the original sentence. For the arguments of the occurrences of the copula from (7a) and (7b) and the verbs from (3) and (1), this is demonstrated in (11), respectively (12):

(11) a. Chris noticed [$_{\text{NP}}$the problem].
b. Chris noticed [$_{\text{CP}}$that Mary hates Bill].

(12) a. The philosopher remembered [$_{\text{NP}}$the discovery].
b. The philosopher remembered [$_{\text{CP}}$that there exists an eighth planet].

Further support for the obtaining of NP/CP equivalences can be found in Appendix B.2.2.

Our expectations on the semantic behavior of proper names in a single-type semantics are summarized in Proposition 1.3:

PROPOSITION 1.3 (Assertoric interpretation of names). *In a single-type semantics, proper names have truth- (and falsity-)conditions* (Prop. 1.3.i), *and are semantically equivalent to some contextually salient sentences* (Prop. 1.3.ii).

The above-cited phenomena illustrate the advantages of interpreting natural language in a single-type semantics (as opposed to traditional Montague semantics). However, the reader is admonished to note that these phenomena can also be accommodated by dropping the assumption of a functional category/type relation from Observation 1.1.ii (Alternative 1), or by explaining the sentence-type behavior of proper names with reference to pragmatics (Alternative 2).[25] The first

[25]I owe this observation to Filip Buekens and Markus Werning.

alternative (adopted in semantic accounts of nonsentential speech, cf. (**Merchant, 2008**; **Culicover and Jackendoff, 2005**; **Dalrymple, 2005**)) assumes that certain occurrences of proper names have a *non-standard semantic content*, which results from 'shifting' the names' standard interpretation (type e) to the standard interpretation of sentences (type $\langle s,t\rangle$). The second alternative (adopted in pragmatic accounts of nonsentential speech, cf. (**Stainton, 2006**; **Borg, 2005**)) assumes that certain utterances of names have a *non-standard asserted content*, which results from attributing names the illocutionary act of making an assertion. Alternative 1 follows the approach of flexible Montague grammar, cf. (**Partee, 1987**; **Hendriks, 1990**). Alternative 2 is inspired by semantic minimalism, cf. (**Borg, 2004**; **Cappelen and Lepore, 2005**). Since this dissertation limits its scope to the domain of semantics, we will exclude Alternative 2 from our further discussion. The possibility of accommodating the phenomena from (1) to (12) in a flexible Montagovian setting (cf. Alternative 1) is discussed below.

Flexible Montague grammar is a variant of traditional Montague semantics which associates every linguistic expression with a *set* of types (rather than with exactly *one* type; cf. Obs. 1.1.ii). Different occurrences of the same expression can then receive an interpretation in the domains of *different* types from this set. For example, instead of interpreting proper names exclusively in the domain of the type e, we can provide their interpretation in the domain of any element from the set $\{e; \langle e,\langle s,t\rangle\rangle; \langle\langle e,\langle s,t\rangle\rangle,\langle s,t\rangle\rangle\}$. Instead of interpreting transitive verbs only in the type $\langle e,\langle e,\langle s,t\rangle\rangle\rangle$, we can give their interpretation in any member of the set $\{\langle e,\langle e,\langle s,t\rangle\rangle\rangle; \langle\langle\langle e,\langle s,t\rangle\rangle,\langle s,t\rangle\rangle,\langle e,\langle s,t\rangle\rangle\rangle; \ldots\}$. These interpretations are obtained from the lowest-rank type of these expressions (for proper names: from the type e) via a number of type-shifting rules.

Type-shifting rules enable the accommodation of some linguistic phenomena which cannot be explained in traditional Montague semantics. In particular, the e-to-$\langle e,\langle s,t\rangle\rangle$ rule *ident* – which enables the interpretation of names in the type for functions from individuals to propositions, $\langle e,\langle s,t\rangle\rangle$ – facilitates the use of proper names as count nouns (in (13); cf. (**Ziff, 1977**, p. 326)): Once it has been lifted to the type $\langle e,\langle s,t\rangle\rangle$, the name Napoleon can combine with the determiner a (type $\langle\langle e,\langle s,t\rangle\rangle,\langle\langle e,\langle s,t\rangle\rangle,\langle s,t\rangle\rangle\rangle$) to form a noun phrase (type $\langle\langle e,\langle s,t\rangle\rangle,\langle s,t\rangle\rangle$). The $\langle\langle e,\langle s,t\rangle\rangle,\langle s,t\rangle\rangle$-to-$\langle e,\langle s,t\rangle\rangle$ rule *Be* explains the possibility of combining NPs (type $\langle\langle e,\langle s,t\rangle\rangle,\langle s,t\rangle\rangle$) with adjective phrases (type $\langle e,\langle s,t\rangle\rangle$) and of coordinating NPs and adjective phrases in the complement of verbs like consider (in (14), resp. (15); cf. (**Partee, 1987**, p. 119)). The latter observations are similar to the observations from (1) to (3), and from (4) to (6):

(13) He is a [$_{\text{NP}}$Napoleon].

(14) a. Mary considers John [$_{\text{AP}}$competent in semantics].

b. Mary considers John [$_{\text{NP}}$an authority on unicorns].

(15) Mary considers John [$_{\text{AP}}$competent in semantics] and [$_{\text{NP}}$an authority on unicorns].

The extension of the set of name-interpretations via the type for propositions, $\langle s,t\rangle$, and the introduction of other type[26]-shifting rules involving propositions (esp. the introduction of e-to-$\langle s,t\rangle$ and $\langle s,t\rangle$-to-e rules) enable the modeling of the phenomena from (1) to (12). Since NP/CP complement-neutral verbs and the copula be can then be shifted to the type $\langle\langle s,t\rangle,\langle e,\langle s,t\rangle\rangle\rangle$, they accommodate the examples from (1) to (3) and (7) (cf. (4)–(6), (11)–(12)). Merchant (**2008**) has shown that the interpretation of proper names in the domain of the type $\langle s,t\rangle$ further enables the modeling of the phenomena from (9) and (10).

The possibility of accommodating all of the above phenomena in a *small* extension of an *existing* generalization of traditional Montague semantics suggests the relative *weakness* of the presented empirical motivation for single-type semantics.[27] In Section 1.4, we will give stronger, methodological, reasons for the adoption of a single-type semantics. However, before we do so, we briefly present formal support for the possibility of a single-type semantics (in Sect. 1.2.2) and identify the objectives of this dissertation (in Sect. 1.3).

1.2.2. Formal Considerations. Clearly, any motivation for single-type semantics is worthless unless we have affirmed the possibility of constructing single-type models. Partee (**2006**) makes a first attempt at undertaking this task. In particular, to provide *formal* support for Proposition 1.2, she identifies a preliminary single-type candidate, i.e. properties of Kratzer-style situations, cf. (**Kratzer, 1989**), and gives its *ad hoc* model for a miniature fragment of English. This model interprets the expressions you, a snake, and see into the single-type objects ⟦you⟧, ⟦a snake⟧, and ⟦see⟧, respectively (**Partee, 2006**, p. 40):

⟦you⟧ the property of (being) a minimal situation containing you;

⟦a snake⟧ the property of (being) a snake-containing situation;

⟦see⟧ a function from two situation-properties p_1 and p_2 to a property p_3 which holds of a situation s_3 if s_3 contains two situations, s_1 and s_2, with the properties p_1, resp. p_2, where (sth. in) s_1 sees (sth. in) s_2.

The above enable the compositional interpretation of the sentence You see a snake:

[26]Since these rules – like Partee's rule *ident* – do not correspond to valid inferences in intuitionistic implicational logic, cf. (**Lambek, 1958**; **van Benthem, 1989**), we hereafter describe them as '*meaning*-shifting rules'. A particular instance of the $\langle s,t\rangle$-to-e rule is used in (**Potts, 2002**).

[27]However, since it obviates the need for e-to-$\langle s,t\rangle$ and $\langle s,t\rangle$-to-e rules, single-type semantics accommodates these phenomena more directly than an extension of flexible Montague semantics.

⟦You see a snake⟧ the property of (being) a situation in which you see a/the snake (which is contained in the situation).

Partee's model supports a type-neutral interpretation of proper names, sentences, and complement phrases. At the same time, it suggests a strategy for the model's extension to larger PTQ-like fragments. However, the nature of Partee's paper (a short *Festschrift* contribution, cf. (**Beck and Gärtner, 2009**)) prevents a demonstration of this extension. Further, since Partee's model is only described informally, and since it only provides a semantics for a very *small* (i.e. four-word) fragment, it does not provide compelling support for Proposition 1.2.

1.3. Objective and Overview

This dissertation formalizes and systematically extends Partee's formal evidence for Proposition 1.2. In particular, it will develop a single-type semantics for the PTQ fragment which provides formal support for Partee's conjecture and which accommodates the semantic behavior of proper names and sentences from Proposition 1.3. The resulting semantics will unify Montague's linguistic ontology, and will yield insight into the apparatus of types in formal semantics.

Our development of a single-type semantics proceeds in correspondence with the conjectures from the previous section in three steps: In particular, Chapters 3, 7, and 8 will present increasingly complex single-type semantics, which accommodate Proposition 1.2, Propositions 1.2 and 1.3.i, and Propositions 1.2, 1.3.i, and 1.3.ii, respectively. As a result, these semantics enable the same-type interpretation, the truth-evaluation, and the identification of equivalent name/sentence pairs (in that order). The single-type semantics from Chapter 8 serves as the 'intended' semantics, which exhibits all desired properties.

The distribution of the presentation of this semantics over different parts of the dissertation enables us to identify the challenges of providing a single-type semantics, and allows an incremental introduction of the core semantic notions. The contribution of the individual chapters to the obtaining of a suitable single-type semantics is discussed below:

To prime the reader's intuitions – and to identify a first set of challenges for the development of a single-type semantics –, Chapter 3 (Part I) provides the simplest possible single-type semantics for the PTQ fragment. This semantics interprets all PTQ expressions into constructions[28] out of the primitive single basic type o from Section 1.2. Since the type o is neutral between Montague's types e and $\langle s, t\rangle$, proper names, sentences, and complement phrases will all receive an interpretation in this type. Our o-based semantics accommodates the phenomena from (1) to (7), and supports Partee's conjecture (Prop. 1.2). However, because of the

[28]Such constructions are obtained through the use of the type-forming rule **ST**.

primitiveness of the type o, the semantics cannot evaluate the truth or falsity of proper names and sentences (cf. Prop. 1.3.i). Since it is further unable to identify a name's sentential equivalents (cf. (9a), (10a); Prop. 1.3.ii), our o-based semantics disqualifies as a suitable single-type semantics for the PTQ fragment.

The remainder of the dissertation attempts to develop an empirically adequate single-type semantics which accommodates the observations from Proposition 1.3. To this aim, we identify the type o with a complex (non-primitive) Montague type. We first identify two suitable Montague types which take the role of the type o from Part I (in Part II). We then define the single-type semantics which are associated with these two types (in Part III).

Specifically, to find a Montague type which satisfies Proposition 1.3, Part II introduces a set of semantic requirements which ensure the type's suitability as a single semantic basis for the PTQ fragment. The application of these requirements to the set of Montague types identifies the types $\langle s, t\rangle$ and $\langle s, \langle s, t\rangle\rangle$ as suitable single-type candidates. These types are associated with propositions and with propositional concepts[29], respectively. The strictness of the requirements on a suitable single basic type motivates the claim from Proposition 1.1.

Chapter 7 (Part III) presents the single-type semantics of the type $\langle s, t\rangle$. This semantics commands a notion of truth for basic-type terms (Prop. 1.3.i) and identifies equivalent name-sentence pairs. The latter is made possible by the characterization of the single basic type as a construction to the truth-value type t, and by the particular strategy for the representation of individuals and propositions. This strategy represents propositions (e.g. 'Barbara Partee is arriving') by themselves (i.e. by the set of indices at which Barbara Partee is arriving), and represents individuals (e.g. Barbara Partee) by the set of indices in which these individuals exist. As a result, proper names (e.g. the name Barbara Partee from (9a)) will be equivalent to their containing simple existential sentences (i.e. Barbara Partee exists). However, because of the comparative informational poverty of propositions, our $\langle s, t\rangle$-based semantics still fails to identify a name's *contextually salient* equivalents (e.g. the sentence Barbara Partee is arriving from (9c); cf. Prop. 1.3.ii). Consequently, this semantics is also excluded as a 'good' single-type semantics for the PTQ fragment.

The single-type semantics from Chapter 8 (Part III) amends the above deficiencies. In this semantics, Proposition 1.3.ii is accommodated by the adoption of the basic type $\langle s, \langle s, t\rangle\rangle$. Objects of this type represent individuals by functions from indices w to the set of indices at which the designators of all w-true propositions which carry information about the individuals are true. Propositions are represented by functions from indices w to the set of indices at which the propo-

[29]The name *propositional concept* has been suggested by Jeroen Groenendijk.

sitions' designators and all w-true propositions about the propositions' type-e arguments are true. In virtue of this representational strategy, it holds that, if the sentence from (9c) is true at an index, the interpretation of the name Barbara Partee at that index is exactly the interpretation of the sentence from (9c) at the index, such that the former is equivalent to the latter.

Some of the material of this dissertation is rather technical. To make its content as accessible as possible, we have taken care to provide all new definitions with a detailed informal motivation and explanation (cf. esp. Part II). Each formal chapter contains a prose description of its philosophical issues and implications. Most proofs have been placed in a separate appendix (Appendix C). Many translations and definitions of logical PTQ forms have been deferred to Appendix D. Appendix A contains a list of abbreviations, notational conventions, and a glossary. Appendix B presents Carstairs-McCarthy's arguments for the formulation of a single-category syntax, cf. (**Carstairs-McCarthy, 1999**; **2005**), and provides further empirical motivations for Partee's conjecture.

We close the chapter by presenting methodological, or philosophical, reasons for the adoption of a single-type semantics. A discussion of the principal percursors of single-type semantics and of alternative approaches to this semantics can be found in Chapter 9.

1.4. Other Motivations for Partee's Conjecture

Methodological reasons for the adoption of a single-type semantics include the complete unification of Montague's semantic ontology (with the expected consequences), the identification of new representability relations between different types of Montagovian objects, and the provision of formal support for Montague's original type system. We discuss these three motivations in their order of mention.

1.4.1. Unification of Types. The interpretation of natural language in a single-type semantics enables a *complete unification* of Montague's semantic ontology: Rather than generating all members of the linguistic zoo from individuals and propositions (cf. Sect. 1.1.2), we will be able to obtain them from a *single* basic type of object.

We expect that the unifying semantics for the PTQ fragment will share the methodological advantages of other unified theories in science. Clearly, in virtue of its ontological parsimony, single-type semantics will be simpler and will be cognitively more economical than Montague semantics. However, we also expect that our single-type semantics will improve upon some of the explanatory power of traditional Montague semantics. This is due to the fact that, in a single-type system, names and sentences will share certain semantic properties. This fact helps us explain the truth-evaluability of proper names, and the obtaining of semantic

equivalence relations between names and sentences (Prop. 1.3; cf. (9), (10)).

By its ability to accommodate these phenomena, single-type semantics will potentially[30] further display a higher degree of evidential support than Montague semantics. This is, in particular, due to the fact that the same-type interpretation of proper names and sentences effects a larger number of probabilistic dependency relations between the semantic correlates of phrase structure rules. To prevent a digression from the central topic of this dissertation, we leave a demonstration of this fact for another occasion. The higher degree of confirmation of a separately-[31] over a Montague-typed semantics is shown in (**Liefke and Hartmann, 2011**).

This completes our discussion of the unificatory power of single-type semantics. The next section emphasizes the ability of single-type semantics to identify representability relations between different types of Montagovian objects. This ability witnesses a weaker kind of unification.

1.4.2. Relations Between Types. The identification of representational relations between different types of objects is one of the most remarkable methodological accomplishments of Montague semantics (cf. Sect. 1.1.2): Since n-th order properties of individuals are modeled as (functions from ...) functions from individuals to sets of indices (to ... sets of indices), they can be used to represent k-th order properties of individuals for every $k \leq n \in \mathbb{N}$. For example, we can thus represent individuals (type e) by type-$\langle e, \langle s, t\rangle\rangle$ functions from individuals to the set of indices in which these individuals exist, by type-$\langle\langle e, \langle s, t\rangle\rangle, \langle s, t\rangle\rangle$ functions from properties of individuals to the set of indices at which the individuals have the relevant property, etc. For the representation of Barbara Partee, the latter are associated with the property of being B. Partee, resp. of being her properties.

Flexible Montague grammar (cf. Sect. 1.2.1) identifies the specific operations which send objects of a lower Montague type (in particular, objects of the types e and $\langle e, \langle s, t\rangle\rangle$) to their representations in a higher type (i.e. in the types $\langle e, \langle s, t\rangle\rangle$, resp. $\langle\langle e, \langle s, t\rangle\rangle, \langle s, t\rangle\rangle$), and vice versa. The domains of some of these operations are given in Figure 1.2 (next page), cf. (**Partee, 1987**). In the figure, we mark inverse images of type-lifting operations by a dashed line.

The operations *ident* and *lift* from Figure 1.2 are the representational operations from the second-to-last paragraph. Thus, the operation *ident* sends individuals to the property of being those individuals. The operation *lift* sends individuals to the property of being a property of those individuals.

---

[30] The outcome of this analysis depends on the number of phenomena which can*not* be explained in single-type semantics. For English, some of these phenomena are discussed in Chapter 3.4.

[31] In a separately-typed semantics, the semantic type system preserves all syntactic distinctions.

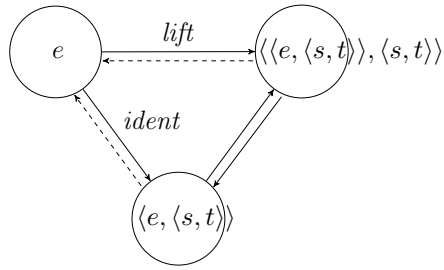

FIGURE 1.2. Representational relations between objects of the types e, $\langle e, \langle s, t\rangle\rangle$, and $\langle\langle e, \langle s, t\rangle\rangle, \langle s, t\rangle\rangle$.

By identifying a single basic type, and by specifying the particular operations which send Montagovian individuals and propositions to objects of the single basic type, single-type semantics *extends* the set of representational relations from flexible Montague grammar. For the particular single-type candidates from Chapters 7 and 8 (here, labelled 'o_1' and 'o_2'), the domains whose elements are connected by these relations are identified in Figure 1.3. Since we expect that basic single-type objects also represent Montagovian propositions, we further add a node for the type $\langle s, t\rangle$. To emphasize the new representational relations, we mark the operations from Figure 1.2 in grey. Inverse images are omitted for perspicuity.

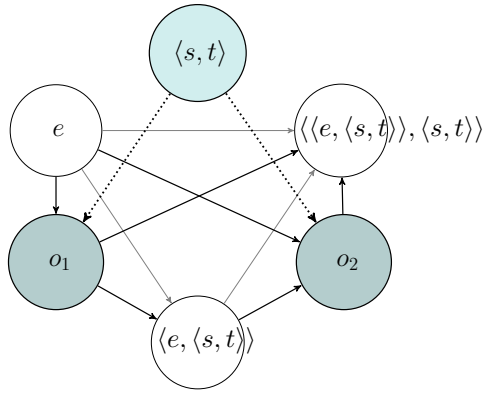

FIGURE 1.3. Relations between objects of the types from Figure 1.2, propositions, and objects of the two single-type types.

We will introduce specific candidates for the e-to-o_1- and e-to-o_2-shifting functions from Figure 1.3 in Chapters 7 and 8, respectively.

1.4.3. Support for Montague Types. The preceding subsections have given the methodological rationale behind our interest in single-type semantics: Since single-type semantics will be (ideally) simpler, more unifying, and better confirmed than traditional Montague semantics, they will be methodologically preferable over this semantics. But the formulation of a single-type semantics also has another, more abstract, motivation.[32] This motivation concerns the identification of the minimal number of types which are required for the interpretation of the PTQ fragment: If the formulation of a single-type semantics succeeds *without* reference to Montagovian typing constraints, we take the semantics to *refute* the common classification of the PTQ fragment as a fragment whose interpretation requires a minimum of *two* semantic types. If the successful formulation of a single-type semantics *relies* on Montagovian typing constraints, we conclude that the semantics *supports* the common classification of the PTQ fragment as a 'two type'-fragment. Thus, the formulation of a single-type semantics will produce support *for* or *against* Montague's original type system.

The small single-type model from (**Partee, 2006**) provides support *for* the Montagovian type system in the above sense. This is due to the fact that Partee's descriptions of single-type objects still make reference to Montagovian objects. Thus, Partee describes the single-type interpretation of the pronoun you as "the property of a minimal situation containing you" (cf. Sect. 1.2.2). But situations (esp. minimal situations) and individuals (e.g. the individual denoted by you in the relevant context) do not qualify as members of Partee's single-type ontology, which consists only of *properties* of situations. As a result, Partee's semantics still generates support for the Montagovian type system. We will show in Parts II and III that our single-type semantics generates similar support.

This concludes our methodological considerations about a single-type semantics. We close this chapter by recommending different orders of reading for different audiences.

1.5. Intended Audience and Order of Reading

The topic of this dissertation lies at the heart of formal semantics: Montague's distinction between individuals and propositions is canonical in every university-level introduction to compositional semantics.[33] The historical predecessor of this distinction, i.e. Frege's separation of objects from truth-values, cf. (**Frege, 1892**), is discussed in every introduction to the philosophy of language.[34] As a result, this

[32]This section was prompted by discussions with Cleo Condoravdi and Seth Yalcin.

[33]See the textbooks (**Chierchia and McConnell-Ginet, 2000**), (**de Swart, 1998**), and (**Heim and Kratzer, 1998**).

[34]The discussions in (**Lycan, 2008**), (**Stainton, 1996**), and (**Martinich and Sosa, 2012**) support this claim.

thesis will be of interest to formal semanticists, and to philosophers of language and of linguistics. However, the previous section has already suggested a much wider scope of this work. Thus, it will also engage the attention of linguistically inclined philosophers of science, logicians, and theoretical computer scientists.

The present section recommends different orders of reading for researchers from these disciplines. This is suggested by their different background knowledge, and by their interests in distinct aspects of the presented theory. For example, while all formal semanticists, most theoretical computer scientists, and many philosophers of linguistics will be familiar with the foundations and applications of Montague semantics, some philosophers of language and many philosophers of science may not (fully) command this knowledge. Further, while members of the former group will have a great interest in the details of the proposed semantics and in its use in the interpretation of a regimented fragment of English, members of the latter group will be primarily interested in the philosophical motivation of the presented theory, and in its general methodological consequences.

The alternative orders of reading for each these audiences are given in Figure 1.4 (next page). In the figure, unbroken arrows indicate the recommended order of reading for formal semanticists, logicians, and computer scientists. Dotted and dashed arrows suggest the order of reading for philosophers of science, and for philosophers of linguistics and of language, respectively. Philosophical (sub-) chapters are represented by light grey boxes. Formal (sub-)chapters are indicated by white boxes.

The introduction to Montague semantics from Chapter 1.1 will be of special interest to philosophers of science, and (depending on their background) to philosophers of linguistics and language, logicians, and computer scientists. It distinguishes itself from existing survey articles on the subject[35] in its focus on the *foundations* of Montague's formal semantics. In virtue of the latter, the chapter will serve as a point of reference for many discussions in this dissertation.

The formal chapters from Parts I and III will be of special interest to formal semanticists, logicians, and computer scientists. On the basis of Chapter 1, philosophers of science, of linguistics, and of language can proceed directly to the informal chapters from Part II. Philosophers of linguistics and language will further be able to jump to the philosophically significant parts of the logical Chapters 2 and 6, and the linguistic Chapters 3, 7, and 8. The conclusion combines all aspects of this work.

[35]Prominent examples include (**Janssen, 2012**), (**Partee, 1997**), and (**Partee, 2001**).

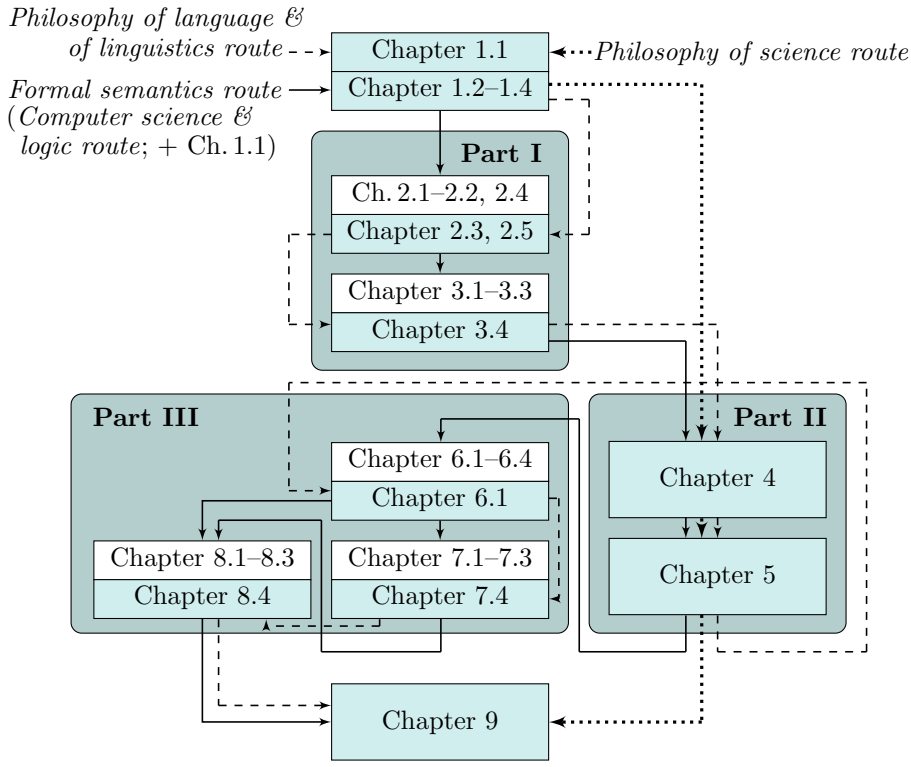

FIGURE 1.4. Alternative orders of reading.

1.6. Sources of Chapters

Part I is based on:

> Liefke, Kristina. 2013a. "A Single-Type Logic for Natural Language." In *Journal of Logic and Computation*, special issue for CiE 2010, edited by Alessandra Carbone, Fernando Ferreira, Benedikt Löwe, and Elvira Mayordomo. [Online first since 01-2013; printed version forthcoming]. http://logcom.oxfordjournals.org/content/early/2013/01/23/logcom.exs074.short.

Chapter 4 is based on:

> Liefke, Kristina. 2013b. "A Single-Type Ontology for Natural Language." In *Was dürfen wir glauben? Was sollen wir tun? Sektionsbeiträge des Achten Internationalen Kongresses der Gesellschaft für Analytische Phi-*

losophie e.V., edited by Miguel Hoeltje, Thomas Spitzley, and Wolfgang Spohn, 70–84. Duisburg-Essen: DuE-Publico. http://duepublico.uni-duisburg-essen.de/servlets/DerivateServlet/Derivate-33085/GAP8_Proceedings.pdf#page=71.

Chapter 7 is based on:

Liefke, Kristina. Forthcoming. "A Single-Type Semantics for the PTQ*-Fragment." Accepted for publication in *Proceedings of Sinn und Bedeutung 18*, edited by Anamaria Fălăuş *et al.* Semanticsarchive.

Some of the ideas behind Chapter 1.4.1 are developed in:

Liefke, Kristina, and Stephan Hartmann. 2011. "Integrative Reduction, Confirmation, and the Syntax-Semantics Map" (Manuscript, Tilburg Center for Logic and Philosophy of Science). *PhilSci Archive*, May 2011. http://philsci-archive.pitt.edu/8760/.

Part I

'Pure' Single-Type Semantics

This part of the dissertation presents the simplest possible single-type semantics in the sense of Proposition 1.2. This semantics is a theory of meaning for natural language which interprets sentences, proper names, and complement phrases in the same unstructured, or '*pure*', semantic type. All objects in this semantics are obtained from basic-type objects via a variant of the type-forming rule **ST**. The semantics interprets all logical forms from Montague's PTQ fragment. Further, it enables the interpretation of NP/CP complement-neutral verbs, name/CP coordinations, and CP equatives. However, as a result of the primitiveness of the single basic type, the semantics cannot evaluate the truth or falsity of proper names and sentences (cf. Prop. 1.3.i), and is unable to identify a name's sentential equivalents (cf. Prop. 1.3.ii).

To provide the described single-type semantics for the PTQ fragment, we use Montague's method of *indirect interpretation* (cf. Fig. 1.1), which proceeds via the compositional translation of logical PTQ forms into the language of the simplest single-type logic, TY_0. The plan for this part is then as follows: Chapter 2 presents a general class of languages and models for the logic TY_0 (cf. step 2). Chapter 3 identifies a designated TY_0 language and model (cf. step 2), provides a theory of syntax for the PTQ fragment (cf. step 1), and gives a set of rules for the TY_0 translation of logical PTQ forms (cf. step 3). The semantic values of these forms are then obtained via the model-theoretic interpretation of their TY_0 translations.

CHAPTER 2

The 'Pure' Single-Type Logic TY_0

We begin by defining the simplest single-type logic TY_0. The latter is a type theory in the spirit of (**Henkin, 1963**), cf. (**Church, 1940**), whose basic type unifies Montague's types for individuals and propositions. The formulation of this logic identifies the challenges for the definition of a single-type logic, and displays its particular syntactic and semantic properties. The chapter is organized as follows: Section 2.1 specifies the types and terms of the logic TY_0. Section 2.2 defines a class of general models for the logic TY_0, which provide a semantics for TY_0 terms. Section 2.3 discusses the role of metatheory in the formulation of this semantics. Section 2.4 defines the metatheoretical relation of entailment on basic-type TY_0 terms, characterizes its proof-theoretic correlate via a sound Gentzen-style sequent calculus, and proves the generalized completeness of this calculus via a Model Existence theorem. To close, we present different strategies for providing a TY_0 truth-definition (in Sect. 2.5).

We begin by defining the types and terms of the logic TY_0.

2.1. Types and Terms

The logic TY_0 has *one* basic type, o. For lack of a better name, we refer to objects of this type as *primitive entities* (or *entities*)[1]. Our choice of this name is motivated by the description of entities as the most general kind of object which subsumes individuals and propositions, cf. (**Chierchia and Turner, 1988**; **Zaefferer, 2007**; **Niles and Pease, 2001**)[2], and by the neutrality of type-o objects between objects of these two types, cf. (**Partee, 2006**). Since entities cannot be obtained from objects of a non-TY_0 type (e.g. a Montague type) through the application of the type-forming rules **CT** or **IT**, we call the logic TY_0 a '*pure*' single-type logic. Different proposals as to the identity of primitive entities will be discussed in Chapter 4 (Part II), and in Part III.

[1] I thank Markus Werning and Dietmar Zaefferer for suggesting this name.

[2] In (**Zaefferer, 2007**), Montagovian individuals are called '*inventities*' (for '*inventory entities*' [of a situation]). In (**Chierchia and Turner, 1988**), propositions and individuals are named '*information units*' and '*urelements*' (or 'the difference between urelements and information units'), respectively.

From the type for entities, o, the types of the logic TY_0 are obtained via an n-ary variant of the type-forming rule **ST** from Chapter 1.2 as follows:

DEFINITION 2.1.1 (TY_0 types). The set $\mathsf{0Type}$ of TY_0 *types* is the smallest set of strings s.t., for $0 \leq n \in \mathbb{N}$, if $\alpha_1, \ldots, \alpha_n \in \mathsf{0Type}$, then $(\alpha_1 \times \cdots \times \alpha_n) \to o \in \mathsf{0Type}$.

Types of the form $(\alpha_1 \times \cdots \times \alpha_n) \to o$ are associated with functions from n-tuples of objects of the types $\alpha_1, \ldots, \alpha_n$ to primitive entities. Such types are correlates of **ST** types of the form $\langle \alpha_1, \langle \ldots, \langle \alpha_n, o \rangle \rangle \rangle$. As a result, the single-type types $\langle o, o \rangle$, $\langle o, \langle o, o \rangle \rangle$, and $\langle \langle o, o \rangle, o \rangle$ from Chapter 1.2 will be replaced by the TY_0 types $o \to o$, $(o \times o) \to o$, and $(o \to o) \to o$, respectively. These types are associated with functions from entities to entities, with functions from ordered pairs of entities to entities, and with functions from (functions from entities to entities) to entities.

Our use of the type-forming rule from Definition 2.1.1 is motivated by the fact that this rule obviates the 'currying' of multi-argument functions to unary functions of a higher type. As a result, this rule yields a simpler single-type semantics.[3] Our use of n-ary functional types follows the practice of (**Montague, 1970a**), (**Orey, 1959**), and (**Tichý, 1982**). Following Tichý, we write $(\alpha_1 \times \cdots \times \alpha_n) \to o$ as $(\alpha_1 \ldots \alpha_n; o)$, and identify the type (o) with o. We will hereafter call o the *basic* (*base*, or *ground*) type of the logic TY_0, and refer to TY_0 types of the form $(\alpha_1 \ldots \alpha_n; o)$ where $n \geq 1$ as *complex* (or *derived*) types.

This completes our specification of the TY_0 type system. We next define a class of languages for the presented system.

A language L for the logic TY_0 is a countable set $\cup_{\alpha \in \mathsf{0Type}} L_\alpha$ of uniquely typed non-logical constants. These include a constant for the *absurd* (*impossible*, or *maximal*) entity, $\textcircled{\perp}$ (cf. Veltman's (**1985**) *improper fact*). For every TY_0 type α, we further assume a countable set $\mathcal{V}_\alpha$ of uniquely typed variables, with '$\cup_{\alpha \in \mathsf{0Type}} \mathcal{V}_\alpha$' abbreviated as '$\mathcal{V}$'. From these basic expressions, we form complex terms inductively with the help of functional application, lambda abstraction, and the non-logical constant $\doteRightarrow$.

DEFINITION 2.1.2 (TY_0 terms). Let $\alpha_1, \ldots, \alpha_n, \beta \in \mathsf{0Type}$. The set T_α of TY_0 *terms* of the type α is then defined as follows:

(i) $L_\alpha, \mathcal{V}_\alpha \subseteq T_\alpha$, $\quad \textcircled{\perp} \in T_o$;
(ii) If $\boldsymbol{B} \in T_{(\beta\alpha_1 \ldots \alpha_n; o)}$ and $\boldsymbol{A} \in T_\beta$, then $(\boldsymbol{B}(\boldsymbol{A})) \in T_{(\alpha_1 \ldots \alpha_n; o)}$;
(iii) If $\boldsymbol{A} \in T_{(\alpha_1 \ldots \alpha_n; o)}$ and $\boldsymbol{x} \in \mathcal{V}_\beta$, then $(\lambda \boldsymbol{x}.\, \boldsymbol{A}) \in T_{(\beta\alpha_1 \ldots \alpha_n; o)}$;
(iv) If $\boldsymbol{B}, \boldsymbol{C} \in T_{\alpha_1}$, then $(\boldsymbol{B} \doteRightarrow \boldsymbol{C}) \in T_o$.

[3] For a discussion of the advantages of n-ary functional or relational type logics, the reader is referred to (**Muskens, 1989**) and (**van Benthem and Doets, 2001**).

Clause (i) identifies all members of L_α and $\mathcal{V}_\alpha$ as TY_0 terms. Clauses (ii) and (iii) identify the results of functional application and lambda abstraction as TY_0 terms. In particular, the application of a type-$(\beta\alpha_1 \ldots \alpha_n; o)$ term $\boldsymbol{B}$ to a type-β term $\boldsymbol{A}$ yields the designator, $(\boldsymbol{B}(\boldsymbol{A}))$ (type $(\alpha_1 \ldots \alpha_n; o)$), of the result of applying the function denoted by $\boldsymbol{B}$ to the referent of $\boldsymbol{A}$. Abstraction over the type-β argument of a type-$(\alpha_1 \ldots \alpha_n; o)$ term $\boldsymbol{A}$ yields the designator, $(\lambda \boldsymbol{x}.\, \boldsymbol{A})$ (type $(\beta\alpha_1 \ldots \alpha_n; o)$), of a function whose application to some type-β argument d returns the value of $\boldsymbol{A}$ with $\boldsymbol{x}$ interpreted as d.[4] In particular, clause (ii) asserts the possibility of forming entity-designators (e.g. the term $(\boldsymbol{walk}(\boldsymbol{bill}))$) out of the designators of primitive entities (here, $\boldsymbol{bill}$) and functions from entities to entities (here, $\boldsymbol{walk}$).

Since we do not identify the entity type o with a particular Montague type (s.t. the logic TY_0 will, in particular, not have a designated type, t, for truth-values), the familiar logical constants for falsum ($\bot$, type t) and logical implication ($\Rightarrow$, type $(\alpha_1\alpha_1; t)$) are not available in the logic TY_0. The non-logical constants $\textcircled{\scriptsize\bot}$ and $\doteq\!\!\Rightarrow$ from clauses (i) and (iv) serve as their single-type stand-ins.[5] In Section 2.2.2, we will introduce a set of meta-level axioms which ensure that $\textcircled{\scriptsize\bot}$ and $\doteq\!\!\Rightarrow$ display the same semantic behavior as the logical constants $\bot$ and $\Rightarrow$.

Since we only stipulate that $\alpha_1 \in \mathsf{0Type}$, clause (iv) describes $\doteq\!\!\Rightarrow$ as a non-uniquely typed constant, which applies to pairs of arguments of all TY_0 types. As a result, it appears that the logic TY_0 is a variant of the polymorphically typed lambda calculus. To avoid the extension of the TY_0 type system by polymorphic types, cf. (**Girard, 1972**; **Reynolds, 1974**) – and the attendant introduction of explicit quantification over type variables –, we assume a *schematic* (or *abbreviatory*) *polymorphism* of types. The latter is a syntactic device whereby a metatheoretical symbol is used to abbreviate a range of (monomorphic) types. Thus, in (iv), the type variable α_1 may be instantiated by any of the elements in $\mathsf{0Type}$. The constant $\doteq\!\!\Rightarrow$ then represents a family, $\{\doteq\!\!\Rightarrow_{(\alpha\,\alpha;o)} \mid \alpha \in \mathsf{0Type}\}$, of *distinct* identical-looking constants, one for each type.[6]

From $\textcircled{\scriptsize\bot}$ and $\doteq\!\!\Rightarrow$, single-type stand-ins of other truth-functional connectives and quantifiers are easily obtained. In particular, the TY_0 proxies for the constants $\top, \forall, =, \neg, \wedge$, and $\Box$ (i.e. $\textcircled{\scriptsize\top}, \bigwedge, \doteq, \urcorner, \barwedge$, and $\boxdot$) are obtained by variants of the definitions from (**Henkin, 1950**).[7] Below, we let $\boldsymbol{x}$, $\boldsymbol{y}$ (or $\boldsymbol{A}$), $\boldsymbol{X}$ ($\boldsymbol{B}$, $\boldsymbol{C}$), and $\boldsymbol{Y}$ be variables (resp. constants) of the type α, $(\alpha; o)$, resp. $((\alpha; o); o)$, where $\alpha \in \mathsf{0Type}$:

---

[4] Particular rules for the behavior of lambda abstracts will be given in Section 2.4.

[5] Thomason (**1980**) and Fox and Lappin (**2004**) use a similar strategy for the introduction of hyperintensional propositional connectives. However, in their logics, the availability of truth-functional connectives enables the formulation of semantic constraints connecting the former with the latter.

[6] A polymorphic single-type semantics will be sketched in Chapter 9.5.2.

[7] We will justify the definitions of these constants in Chapter 7.2.2 (cf. Nota. 7.2.1 and (7.2.1)).

NOTATION 2.1.1. We write

$$
\begin{array}{rcl}
\top & \text{for} & (\bot \doteRightarrow \bot) \\
\left(\bigwedge \boldsymbol{x}.\boldsymbol{A}\right) & \text{for} & \left((\lambda \boldsymbol{x}.\top) \doteRightarrow (\lambda \boldsymbol{x}.\boldsymbol{A})\right) \\
\boldsymbol{B} \doteq \boldsymbol{C} & \text{for} & \left(\bigwedge \boldsymbol{Y}.\boldsymbol{Y}(\boldsymbol{B}) \doteRightarrow \boldsymbol{Y}(\boldsymbol{C})\right) \\
\sim \boldsymbol{B} & \text{for} & \left(\lambda \boldsymbol{x}.\boldsymbol{B}(\boldsymbol{x}) \doteq \bot\right) \\
(\boldsymbol{B} \wedge \boldsymbol{C}) & \text{for} & \left(\lambda \boldsymbol{x}.(\lambda \boldsymbol{X}.\boldsymbol{X}(\boldsymbol{B} \doteq \boldsymbol{C})) \doteq (\lambda \boldsymbol{X}.\boldsymbol{X}(\top))\right) \\
(\boldsymbol{B} \vee \boldsymbol{C}) & \text{for} & \sim(\sim \boldsymbol{B} \wedge \sim \boldsymbol{C}) \\
\Box \boldsymbol{A} & \text{for} & \left(\bigwedge \boldsymbol{x}.\boldsymbol{x} \doteq \bot \vee (\boldsymbol{x} \doteRightarrow \boldsymbol{A})\right)
\end{array}
$$

The single-type stand-ins, $\not\doteq$, $\dot{\rightarrow}$, $\dot{\leftrightarrow}$, $\dot{\bigvee}$, and $\dot{\Diamond}$, of the familiar symbols for inequality, material implication and biimplication, the existential quantifier, and the modal diamond operator have their expected definitions.

In the above, the letter 'L' ranges over TY_0 languages. In Chapter 3, we will introduce its calligraphic version, '$\mathcal{L}$', as the designator of the particular TY_0 language which translates Logical Form-constituents of the PTQ fragment.

For notational convenience, we will sometimes subscript TY_0 terms by their logical type. Thus, '$\boldsymbol{A}_\alpha$' indicates that $\boldsymbol{A}$ is a TY_0 term of the type α. We take binary connectives to dominate over unary connectives[8], and adopt the usual conventions regarding binding, freedom, and closure. Substitution is defined as usual:

DEFINITION 2.1.3 (Substitution). A *substitution* is a function $\sigma : \mathcal{V}_\alpha \rightarrow \cup_{\alpha \in \mathsf{0Type}} T_\alpha$ which sends TY_0 variables to same-type TY_0 terms. We denote substitutions σ', where $\sigma'(\boldsymbol{x}) = \boldsymbol{A}$ and $\sigma'(\boldsymbol{y}) = \sigma(\boldsymbol{y})$ for all $\boldsymbol{y} \neq \boldsymbol{x}$ by $\sigma[\boldsymbol{x} := \boldsymbol{A}]$. A substitution mapping $\boldsymbol{x}_i$ to $\boldsymbol{A}_i$ and $\boldsymbol{y}$ to $\boldsymbol{y}$ (where $\boldsymbol{y} \notin \{\boldsymbol{x}_i, \ldots, \boldsymbol{x}_n\}$) is written as $\{\boldsymbol{x}_1 := \boldsymbol{A}_1, \ldots, \boldsymbol{x}_n := \boldsymbol{A}_n\}$.

In the definition of substitution, the symbols $=$ and $\neq$ are metalanguage connectives. As a result, expressions of the form $\boldsymbol{y} = \boldsymbol{x}$ or $\boldsymbol{y} \neq \boldsymbol{x}$ do not qualify as terms of the logic TY_0. The metatheory of TY_0 will be sketched in Section 2.3.

This completes our specification of TY_0 types and terms. We next turn to the presentation of TY_0 semantics.

2.2. Models

To provide the logic TY_0 with a semantics, we first define a class of models which enable the interpretation of all TY_0 terms from Definition 2.1.2 and Notation 2.1.1 (in Sect. 2.2.1). To ensure that the designated TY_0 constants $\bot$, $\doteRightarrow$, etc. behave as advertised, we then impose several constraints on these models (in Sect. 2.2.2).

[8]Thus, we read the term $\sim \boldsymbol{A} \wedge \boldsymbol{B}$ as $(\sim \boldsymbol{A} \wedge \boldsymbol{B})$, and not as $\sim (\boldsymbol{A} \wedge \boldsymbol{B})$.

2.2.1. General Models. A general model for the logic TY_0 consists of a general TY_0 frame F, an interpretation function I_F for F, and a variable assignment g_F. In particular, general frames F are defined as hierarchies of typed TY_0 domains, where $\mathcal{O}$ is the set of primitive entities (type o):

DEFINITION 2.2.1 (General TY_0 frames). A *general frame* for the logic TY_0 is a set $F = \{D_\alpha \mid \alpha \in \mathsf{0Type}\}$ of pairwise disjoint non-empty sets such that $D_{(\alpha_1 \ldots \alpha_n; o)} \subseteq \{f \mid f : (D_{\alpha_1} \times \cdots \times D_{\alpha_n}) \to \mathcal{O}\}$ for all TY_0 types $(\alpha_1 \ldots \alpha_n; o)$.

Above, the letter 'F' ranges over general TY_0 frames. We will later introduce the letter '$\mathcal{F}$' as the designator of the particular TY_0 frame whose elements are associated with the TY_0 interpretations of logical forms from the PTQ fragment.

In line with our description of the type o, we call the set $\mathcal{O}$ the *basic* (*base*, or *ground*) domain of the logic TY_0. Sets $D_{(\alpha_1 \ldots \alpha_n; o)}$ where $n \geq 1$ will be called *complex* (or *derived*) domains. We identify $\mathcal{O}$ with the union of the set of *possible* entities and the *absurd entity*. The absurd entity is the value of the TY_0 constant $\textcircled{\scriptsize\perp}$. Possible entities are the values of the constants in the set $L_o \setminus \{\textcircled{\scriptsize\perp}\}$. The set of possible entities includes the *trivial* (*necessary*, or *minimal*) *entity*, which is denoted by the TY_0 constant $\textcircled{\scriptsize\top}$.

To ensure the recursive axiomatizability of the entailment relation – and the attendant completeness of the logic TY_0 –, we associate complex TY_0 domains with subsets of function spaces. We call a frame *general* if every set $D_{(\alpha_1 \ldots \alpha_n; o)}$ is a subset of its associated function space. Our use of general frames follows the strategy from (**Henkin, 1950**).

The relation between TY_0 terms in the language L and their associated objects in a TY_0 frame F is established by means of an interpretation function I_F and a variable assignment g_F. Functions I_F are defined as follows:

DEFINITION 2.2.2 (TY_0 Interpretation). An *interpretation function* $I_F : L \to F$ for a TY_0 language L and frame F assigns to each non-logical constant $\boldsymbol{c}_\alpha$ a type-identical denotation in F, such that $I_F(\boldsymbol{c}_\alpha) \in D_\alpha$.

Variable assignments $g_F : \mathcal{V} \to F$ are analogously defined. Given an object $d \in D_\alpha$ and variables $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{V}_\alpha$, we define the assigment $g_F[d/\boldsymbol{x}]$ by letting $g_F[d/\boldsymbol{x}](\boldsymbol{x}) = d$ and $g_F[d/\boldsymbol{x}](\boldsymbol{y}) = g_F(\boldsymbol{y})$ if $\boldsymbol{x} \neq \boldsymbol{y}$. The set of all assignments g_F with respect to a given TY_0 frame F is denoted by '$\mathcal{G}_F$'.

In Definition 2.2.2, the letter 'I_F' ranges over TY_0 interpretation functions. We will later associate '$\mathcal{I}_\mathcal{F}$' with the particular function which sends constants in the language $\mathcal{L}$ to their semantic values in the designated TY_0 frame $\mathcal{F}$.

The value, $I(\boldsymbol{c})$ (or $g(\boldsymbol{x})$), of a TY_0 term $\boldsymbol{c}$ (resp. $\boldsymbol{x}$) is called the *object* denoted by $\boldsymbol{c}$ (or $\boldsymbol{x}$). Depending on their type, we distinguish two different kinds of single-type objects: *entities* (type o) and *properties* of (or *relations* between) entities

(type $(\alpha_1 \dots \alpha_n; o)$, with $n \geq 1$). In Chapter 3, entities will serve as the interpretations of proper names, sentences, and complement phrases. Entity properties will serve as the interpretations of logical forms of expressions from the remaining syntactic categories.

The above considerations have prepared the definition of general TY_0 models. However, before we can attend to this task, we first need to specify a way of converting n-ary functions into unary functions. This is required by the polyadic character of our TY_0 type-forming rule (cf. Def. 2.1.1), and the resulting restriction of TY_0 functions of the type $(\alpha_1 \dots \alpha_{\underline{n}}; o)$ to $\underline{n}$-tuples of suitably typed arguments.

Slice functions, cf. (**Muskens, 1995**), allow the application of n-ary functions to a *single* argument. To facilitate their definition, we represent n-ary functions in the domain of the type $(\alpha_1 \dots \alpha_n; o)$ via sets of ordered $n+1$-tuples of the form $\langle d_1, \dots, d_n, d_{n+1} \rangle$, where $d_i \in D_{\alpha_i}$ for each $i \in \mathbb{N}$, and where $d_{n+1} \in \mathcal{O}$.

Let f be a function of the type $(\alpha_1 \dots \alpha_n; o)$ and let $1 \leq k \leq n \in \mathbb{N}$. Slice functions code n-ary functions into unary functions of a higher type as follows:

Definition 2.2.3 (Slice functions). The k-th *slice function* $\mathscr{S}^k_f$ of the function f applies to members of the set D_{α_k} to yield $n-1$-ary functions in the domain of the type $(\alpha_1 \dots \alpha_{k-1}\, \alpha_{k+1} \dots \alpha_n; o)$.

Thus, the application of the slice function $\mathscr{S}^k_f$ to a type-α_k object d fixes the k-th argument place of f to d:

$$\mathscr{S}^k_f(d) = \big\{ \langle d_1, \dots, d_{k-1}, d_{k+1}, \dots, d_n, d_{n+1} \rangle \mid f(d_1, \dots, d_{k-1}, d, d_{k+1}, \dots, d_n) = d_{n+1} \big\} \tag{2.2.1}$$

On the basis of the above, we can then define general TY_0 models as follows:

Definition 2.2.4 (General TY_0 models). A *general model* for the logic TY_0 is a triple $M_F = \langle F, I_F, V_F \rangle$, consisting of a general TY_0 frame F, an interpretation function I_F, and a function $V_F : (\mathcal{G}_F \times \cup_\alpha T_\alpha) \to F$. The function V_F assigns to every assignment g_F and non-logical term $\boldsymbol{A}_\alpha$ an object in the domain D_α s.t.

(i) $V_F(g_F, \boldsymbol{c}) := I_F(\boldsymbol{c})$ if $\boldsymbol{c} \in L$,
$V_F(g_F, \boldsymbol{x}) := g_F(\boldsymbol{x})$ if $\boldsymbol{x} \in \mathcal{V}$;

(ii) $V_F(g_F, \boldsymbol{B}(\boldsymbol{A})) := \mathscr{S}^1_{(V_F(g_F, \boldsymbol{B}))}\big(V_F(g_F, \boldsymbol{A})\big)$;

(iii) $V_F(g_F, \lambda \boldsymbol{x}_\beta . \boldsymbol{A}) := \big\{ \big\langle d, V_F(g_F[d/\boldsymbol{x}], \boldsymbol{A}) \big\rangle \mid d \in D_\beta \big\}$.

In the above, clause (i) summarizes the characterization of the interpretation and assignment functions from Definition 2.2.2. Since we have identified the single-type stand-ins for falsum and logical implication as members of L, the first item of clause (i) also defines the results of applying V_F to the pairs $\langle g_F, ⓘ \rangle$ and $\langle g_F, \doteq \rangle$. Clauses (ii) and (iii) define the interpretation of application and lambda abstrac-

tion. In particular, the function V_F interprets abstraction over the type-β argument of the term $\boldsymbol{A}$ from Definition 2.1.2.iii as the function whose application to some object $d \in D_\beta$ yields the type-$(\alpha_1 \ldots \alpha_n; o)$ object $V_F(g_F[d/\boldsymbol{x}], \boldsymbol{A})$. To enable the application of the type-$(\beta\alpha_1 \ldots \alpha_n; o)$ function $V_F(g_F, \boldsymbol{B})$ to the type-β object $V_F(g_F, \boldsymbol{A})$ (rather than to an ordered n-tuple of objects of the types $\beta, \alpha_1, \ldots, \alpha_n$), the interpretation of the term $\boldsymbol{B}(\boldsymbol{A})$ from Definition 2.1.2.ii uses the first slice function of the function $V_F(g_F, \boldsymbol{B})$.

This completes our discussion of general models for the logic TY_0. To ensure that the designated single-type constants from Definition 2.1.2 and Notation 2.1.1 emulate the semantic behavior of their associated connectives and quantifiers, we next introduce a set of non-logical meta-level axioms for these constants.

2.2.2. Constraints on Models. Axioms **Ax1** to **Ax9** (below) refer to predicate-logical formulas (in particular, to formulas of the TY_0 metatheory; cf. Sect. 2.3). In the axioms, $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ are TY_0 terms of the same suitable type. We assume that $\boldsymbol{X}$ is a set of type-identical TY_0 terms.

Ax1. $(\boldsymbol{A} \wedge \boldsymbol{A}) = \boldsymbol{A}$, and $(\boldsymbol{A} \vee \boldsymbol{A}) = \boldsymbol{A}$

Ax2. $(\boldsymbol{A} \wedge \boldsymbol{B}) = (\boldsymbol{B} \wedge \boldsymbol{A})$, and $(\boldsymbol{A} \vee \boldsymbol{B}) = (\boldsymbol{B} \vee \boldsymbol{A})$

Ax3. $((\boldsymbol{A} \wedge \boldsymbol{B}) \wedge \boldsymbol{C}) = (\boldsymbol{A} \wedge (\boldsymbol{B} \wedge \boldsymbol{C}))$, and $((\boldsymbol{A} \vee \boldsymbol{B}) \vee \boldsymbol{C}) = (\boldsymbol{A} \vee (\boldsymbol{B} \vee \boldsymbol{C}))$

Ax4. $(\boldsymbol{A} \wedge (\boldsymbol{A} \vee \boldsymbol{B})) = \boldsymbol{A}$, and $(\boldsymbol{A} \vee (\boldsymbol{A} \wedge \boldsymbol{B})) = \boldsymbol{A}$

Ax5. $(\boldsymbol{A} \wedge (\boldsymbol{B} \vee \boldsymbol{C})) = ((\boldsymbol{A} \wedge \boldsymbol{B}) \vee (\boldsymbol{A} \wedge \boldsymbol{C}))$, and
$(\boldsymbol{A} \vee (\boldsymbol{B} \wedge \boldsymbol{C})) = ((\boldsymbol{A} \vee \boldsymbol{B}) \wedge (\boldsymbol{A} \vee \boldsymbol{C}))$

Ax6. $(\textcircled{\scriptsize$\bot$} \wedge \boldsymbol{A}) = \textcircled{\scriptsize$\bot$}$, and $(\textcircled{\scriptsize$\top$} \vee \boldsymbol{A}) = \textcircled{\scriptsize$\top$}$

Ax7. $\neg\big(\bigwedge_{\boldsymbol{A} \in \boldsymbol{X}} \boldsymbol{A}\big) = \big(\bigvee_{\boldsymbol{A} \in \boldsymbol{X}} \neg \boldsymbol{A}\big)$, and $\neg\big(\bigvee_{\boldsymbol{A} \in \boldsymbol{X}} \boldsymbol{A}\big) = \big(\bigwedge_{\boldsymbol{A} \in \boldsymbol{X}} \neg \boldsymbol{A}\big)$

Ax8. $\neg\neg \boldsymbol{A} = \boldsymbol{A}$

Ax9. $\neg\textcircled{\scriptsize$\bot$} = \textcircled{\scriptsize$\top$}$, and $\neg\textcircled{\scriptsize$\top$} = \textcircled{\scriptsize$\bot$}$

The above axioms induce on the set of entities the behavior of a complete De Morgan algebra. In particular, since we have defined the constants $\wedge$ and $\vee$ from the primitive constants $\doteq\!\!\Rightarrow$ and $\textcircled{\scriptsize$\bot$}$ (cf. Nota. 2.1.1), axioms **Ax1** to **Ax4** attribute to the TY_0 proxy for logical implication, $\doteq\!\!\Rightarrow$, the properties of a partial ordering. Axioms **Ax6** and **Ax9** make $\textcircled{\scriptsize$\top$}$ and $\textcircled{\scriptsize$\bot$}$ behave like the algebra's top and bottom elements. Axioms **Ax5**, **Ax7**, and **Ax8** characterize the single-type proxy for negation as De Morgan involution.

Our characterization of the set of entities as a complete De Morgan algebra will later be motivated by our semantic analysis of the type o from Chapter 4 (Part II), and by the algebraic properties of the single-type logics from Part III. However, since we associate entities with the semantic values of *sentences*, we can provide many of the above axioms with an independent rationale.

In particular, the obtaining of **Ax1** through **Ax4** is warranted by the need to capture entailment relations between natural language sentences in a TY_0 semantics: A competent speaker of a language can judge, for any two sentences S and S', whether S entails S', whether S' entails S, or whether S and S' fail to be connected by entailment. These relations hold if S contains *at least* the semantic content of S' (s.t. S is the result of *combining* the content of S' with the content of some other sentence), if S contains *at most* the semantic content of S' (s.t. S is the result of *joining* the content of S' with the content of some other sentence), and if neither S nor S' contains the semantic content of S', resp. S.

To enable the formal modeling of linguistic entailment, we assume that entities are also ordered with respect to their semantic content. We identify the trivial entity and the absurd entity with the semantically 'poorest' and the semantically 'richest' (or *contradictory*) element in this ordering, respectively (**Ax6**). The contradictoriness of the absurd entity is understood as the result of combining 'too much' (eventually conflicting) semantic content. We interpret $(\boldsymbol{A} \doteqdot \boldsymbol{B})$ as asserting that the entity denoted by $\boldsymbol{A}$ *contains the semantic content of* the entity denoted by $\boldsymbol{B}$. Correspondingly, we let $(\boldsymbol{A} \wedge \boldsymbol{B})$ denote the *poorest* entity which contains the content of both $\boldsymbol{A}$ *and* $\boldsymbol{B}$, and let $(\boldsymbol{A} \vee \boldsymbol{B})$ denote the *richest* entity which contains the content of $\boldsymbol{A}$ *or* $\boldsymbol{B}$.

The structure on entities is depicted with the help of a Hasse diagram in Figure 2.1:

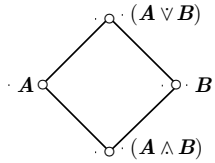

FIGURE 2.1. An entity algebra.

Notably, the semantic meet- and join-operations $\wedge$ and $\vee$ are not exhaustively defined by axioms **Ax1** to **Ax4**. Their behavior is further governed by the laws of Distributivity (**Ax5**) and of Top and Bottom (**Ax6**). Axiom **Ax6** follows trivially from the existence of a semantically poorest ($\top$) and a richest entity ($\bot$). The distributivity of $\mathcal{O}$ holds since, intuitively, none of its subalgebras is a pentagon or a diamond (i.e. a set with one of the structures from Fig. 2.2, next page).

However, the resulting description of the set of entities is still not rich enough for our purposes. It remains to characterize the behavior of TY_0 negation. This operation is associated with complementation.

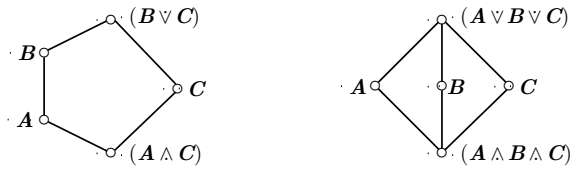

FIGURE 2.2. A pentagon (left) and a diamond (right).

Complementation of entities corresponds to the algebra's 180-degree turn, such that the algebra's top and bottom elements will be represented by the absurd, respectively by the trivial entity (**Ax9**). For reasons having to do with our particular semantic analysis of entities (cf. Part II, Ch. 4), we do not adopt axioms for consistency (($\boldsymbol{A} \wedge \neg \boldsymbol{A}) = \textcircled{\scriptsize\bot}$) or Excluded Middle (($\boldsymbol{A} \vee \neg \boldsymbol{A}) = \textcircled{\scriptsize\top}$; abbreviated 'LEM'). To obtain a *weaker* notion of complementation, we replace these two axioms by the De Morgan laws (here, the *complete* De Morgan laws; **Ax7**) and the law of Double Negation (**Ax8**). The completeness of the entity algebra is reflected in the possibility of interpreting the conjunction and disjunction of any pair of natural language sentences. The uniqueness of negation is ensured by the distributivity of semantic meet and join (**Ax5**).

This completes our motivation of the non-logical axioms for the TY_0 constants $\wedge$, $\vee$, $\neg$, $\textcircled{\scriptsize\top}$, and $\textcircled{\scriptsize\bot}$. Their use in the definitions of the constants $\doteq$, $\dot{\rightarrow}$, $\dot{\leftrightarrow}$, $\bigwedge$, $\bigvee$, $\boxdot$, and $\diamondsuit$ (cf. Nota. 2.1.1) also constrains the behavior of these constants.

Since we have identified the type o as the *only* basic type of the logic TY_0, and since we have defined TY_0 types as constructions to the type o (Def. 2.1.1), we know that all TY_0 domains inherit the algebraic structure on the set of entities. This observation is captured below:

THEOREM 2.1. *Every* TY_0 *domain is a complete De Morgan algebra.*

PROOF. The proof is analogous to the one in (**Landman, 1991**, p. 285–287).

On the basis of the above, we identify TY_0 models with algebraic (or De Morgan) models. These models are defined as follows:

DEFINITION 2.2.5 (Algebraic TY_0 models). An *algebraic model* (or a *De Morgan model*) for the logic TY_0 is a general model M_F for TY_0 where every domain in the frame F is the carrier of a complete De Morgan algebra.

This ends our discussion of the constraints on TY_0 models. In Section 2.4, we will provide a definition of TY_0 entailment. However, before we do so, we briefly note a difficulty regarding the truth-evaluation of basic TY_0 terms, and emphasize the role of metatheory in the provision of a TY_0 semantics. Different strategies for the provision of a TY_0 truth-definition will be presented in Section 2.5.

2.3. The Role of Metatheory

We have seen in Section 2.1 that the familiar truth-functional connectives and quantifiers are not available in the logic TY_0. To compensate for this shortcoming, we have introduced a set of designated non-logical TY_0 constants (in Def. 2.1.2, Nota. 2.1.1), and have enforced their desired behavior through the use of non-logical axioms (in Sect. 2.2.2).

Since the logic TY_0 lacks a type for truth-values[9] – such that TY_0 terms defy an easy truth-evaluation – we are unable to constrain the set of TY_0 models in the object theory. We solve this problem by constraining the set instead in the logic's *meta*theory. The metatheoretical characterization of logical models is quite common: see (**Smith, 1984**; **Turner, 1992**; **Fox and Lappin, 2004**). For the particular case of the logic TY_0, this method involves the formulation of a set of axioms which restrict the class of TY_0 models to models satisfying these axioms in the metatheory. We identify this theory with an extension of the logic TY_0 via the type for truth-values t.[10] As a result, models of this logic can reference concepts (like truth and entailment), which are not available in the TY_0 object theory.

For the present purposes, we need not worry about the particular properties of the TY_0 metatheory. It suffices to know that the behavior of the designated TY_0 constants $\textcircled{1}$, $\doteq\!\!\Rightarrow$, etc. will be preserved in this theory. Following the specification of the semantic content of entities (cf. Ch. 4, Part II), we will give a detailed presentation of our single-type metatheory in Chapter 6 (Part III).

On the basis of the above, we can pursue our definition of TY_0 semantics.

2.4. Entailment and Proof Theory

The De Morgan algebra on the set of entities induces an entailment structure on TY_0 domains. However, since entailment is a relation of the type $(\alpha\,\alpha;\underline{t})$ (where $\alpha \in \mathsf{0Type}$), the notions of TY_0 entailment and equivalence require a definition in the TY_0 metatheory. Specifically, TY_0 entailment is defined through the partial ordering, $\subseteq$, on TY_0 domains as follows:

Definition 2.4.1 (TY_0 entailment). A set of TY_0 terms $\Gamma = \{\gamma \,|\, \gamma \in T_o\}$ *entails* a set of TY_0 terms $\Delta = \{\delta \,|\, \delta \in T_o\}$ in the TY_0 metatheory, i.e. $\Gamma \models_g \Delta$, if, for all general TY_0 models M_F and assignments g_F,

$$V_F\big(g_F, \bigwedge_{\gamma\in\Gamma} \gamma\big) \subseteq V_F\big(g_F, \bigvee_{\delta\in\Delta} \delta\big).$$

[9] This holds unless $o := t$, which we will exclude in Part II, Chapter 4.2.2.

[10] As a result, the TY_0 metatheory will be an n-ary variant of Church's Simple Theory of Types, where the type for propositions is interpreted as t, and where the type for individuals, ι, is replaced by our type o. Types of this logic then have the form $(\alpha_1 \ldots \alpha_n; o)$ or $(\alpha_1 \ldots \alpha_n; t)$. The logic's language and models are defined in analogy to the relevant concepts of the object theory.

According to Definition 2.4.1, the set Γ entails Δ iff the combination of the semantic values of all terms in the set Γ is included in the union of the values of all terms in the set Δ under the ordering $\subseteq$.

The easy definability of TY_0 entailment (vis-à-vis truth) supports Partee's observation about the centrality of entailment relations in single-type semantics, cf. (**Partee, 2006**, p. 40).

Definition 2.4.1 enables the definition of TY_0 equivalence in terms of mutual entailment:

Definition 2.4.2 (TY_0 equivalence). A TY_0 term $\boldsymbol{A}_o$ is *equivalent* to a TY_0 term $\boldsymbol{B}_o$ in the metatheory, i.e. $\models_g \boldsymbol{A} = \boldsymbol{B}$, if $\boldsymbol{A} \models_g \boldsymbol{B}$ and $\boldsymbol{B} \models_g \boldsymbol{A}$ with respect to all general TY_0 models and assignments.

In the above definition, the collapse of mutual equivalence to semantic (metatheoretical) identity is warranted by the antisymmetry of the relation $\subseteq$.

In the above, the subscript 'g' of the entailment relation refers to the generality of TY_0 models (cf. Def. 2.2.4) and the attendant recursive axiomatizability of the metalogical entailment relation. We call a TY_0 term γ *g-valid* if $\models_g \gamma$ in the metatheory with respect to every *general* TY_0 model and assignment.

To enable a proof-theoretic characterization of TY_0 entailment, we use the symbol for metalogical implication, $\Rightarrow$. Its use enables the formulation of the following deduction theorem. In the theorem, we assume that $\Gamma = \left\{ \bigwedge_{\gamma \in \Gamma} \gamma \right\}$ and $\Delta = \left\{ \bigvee_{\delta \in \Delta} \delta \right\}$ are sets of basic TY_0 terms:

Theorem 2.2 (Deduction theorem for TY_0). *The set* Γ *entails the set* Δ *if* Δ *is deducible from* Γ:

$$\Gamma \models_g \Delta \quad \text{if} \quad \models_g \Gamma \Rightarrow \Delta$$

Proof. The proof, which proceeds inside the TY_0 metatheory, is standard.

An expression which asserts the deducibility of a set of conclusions from a set of premises is called a *sequent*. We characterize the proof-theoretic correlate, $\Rightarrow$, of $\models_g$ via a Gentzen-style sequent calculus. Sequents take the general form of (2.4.1), where Γ_i and Δ_i are sets of basic-type terms for all $i \in \{1, \ldots, n+1\}$.

$$\frac{\Gamma_1 \Rightarrow \Delta_1 \ldots \Gamma_n \Rightarrow \Delta_n}{\Gamma_{n+1} \Rightarrow \Delta_{n+1}} \tag{2.4.1}$$

The fraction notation in (2.4.1) is interpreted as a conditional. Thus, the sequents $\Gamma_1 \Rightarrow \Delta_1, \ldots, \Gamma_n \Rightarrow \Delta_n$ are the antecedent conditions of the sequent rule in (2.4.1); the sequent $\Gamma_{n+1} \Rightarrow \Delta_{n+1}$ is the consequent of this rule. If $n = 0$, the set of a rule's conditions is called *empty*, and the relevant rule *axiomatic*. In Table 2.1 (next page), R is an axiomatic rule.

We next provide some terminology: A sequent $\Gamma \Rightarrow \Delta$ is TY_0-*provable*, i.e. $\Gamma \vdash_{\mathrm{TY}_0} \Delta$, if there are finite sets Γ_0 and Δ_0, with $\Gamma_0 \subseteq \Gamma$ and $\Delta_0 \subseteq \Delta$, such that Δ_0 is deducible from Γ_0, where Γ_0 is a sequent rule or follows by a rule from a term occurring earlier in the proof.

A TY_0 model M_F for the language L *refutes* a sequent $\Gamma \Rightarrow \Delta$ of L if $\models_g \gamma$ for all $\gamma \in \Gamma$ and $\not\models_g \delta$ for all $\delta \in \Delta$ in M_F. A sequent is *g-valid* if it is not refuted by any model.

For convenience, we will often drop the 'TY_0'-subscript of the provability relation and brackets '{', '}'. We abbreviate sets of TY_0 terms by capital Greek letters, and let $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, and $\boldsymbol{x}$ be TY_0 constants, resp. a TY_0 variable of suitable type. Table 2.1 provides the structural rules for the logic TY_0:

$$\frac{}{\boldsymbol{A} \Rightarrow \boldsymbol{A}}\, R \qquad \frac{\Gamma, \boldsymbol{A} \Rightarrow \Delta \qquad \Gamma \Rightarrow \Delta, \boldsymbol{A}}{\Gamma \Rightarrow \Delta}\, \mathsf{cut}$$

$$\frac{\Gamma \Rightarrow \Delta}{\Gamma, \boldsymbol{A} \Rightarrow \Delta}\, W\mathsf{L} \qquad \frac{\Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \boldsymbol{A}}\, W\mathsf{R}$$

TABLE 2.1. Structural rules for TY_0.

The introduction rules for the TY_0 constants $\textcircled{1}$, $\doteq$, $\bigwedge$, $\doteq$, $\lceil$, $\wedge$, $\boxdot$, and $\diamond$ are given in Table 2.2 (next page).

Notably, the logic TY_0 behaves 'rather' classically. Its only difference with respect to classical type theory (**Church, 1940**), cf. (**Henkin, 1950**; **Gallin, 1975**), regards the unavailability of the left introduction rule for Boolean negation (in (2.4.2)). This is due to the absence of LEM in the metalevel axioms for $\lceil$ (cf. Sect. 2.2.2).

$$\frac{\Gamma \Rightarrow \Delta, \boldsymbol{A}}{\Gamma, \lceil \boldsymbol{A} \Rightarrow \Delta}\, \mathrm{Bool}.\lceil\mathsf{L} \tag{2.4.2}$$

To provide a suitable syntactic characterization of the relation of logical consequence, we replace the rule from (2.4.2) by the weaker pair of rules $\lceil\mathsf{L}$ and $\lceil\mathsf{R}$ from Table 2.2. These rules are variants of the negation rules from (**Blamey, 1986**).

The rules ext, $\lambda\mathsf{L}$, and $\lambda\mathsf{R}$ assert the identity of semantically indiscernible objects (*Extensionality*) and the substitutability of $\beta\eta$-equivalent TY_0 terms. From the rules $\lambda\mathsf{L}$ and $\lambda\mathsf{R}$, we can derive the usual rules of lambda conversion:

$$\frac{}{\lambda \boldsymbol{x}.\, \boldsymbol{A} \Rightarrow \lambda \boldsymbol{y}.\, \boldsymbol{A}\{\boldsymbol{x} := \boldsymbol{y}\}}\, \alpha$$
$$\text{if } \boldsymbol{y} \text{ is free for } \boldsymbol{x} \text{ in } \boldsymbol{A}$$

$$\frac{}{(\lambda \boldsymbol{x}.\, \boldsymbol{A})(\boldsymbol{B}) \Rightarrow \boldsymbol{A}\{\boldsymbol{x} := \boldsymbol{B}\}}\ \beta \quad \text{if } \boldsymbol{x} \text{ is free for } \boldsymbol{B} \text{ in } \boldsymbol{A} \qquad \frac{}{\lambda \boldsymbol{x}.\, \boldsymbol{A}(\boldsymbol{x}) \Rightarrow \boldsymbol{A}}\ \eta \quad \text{if } \boldsymbol{x} \text{ is not free in } \boldsymbol{A}.$$

The rules α, β, and η allow the renaming of bound variables (here, the renaming of $\boldsymbol{x}$ as '$\boldsymbol{y}$'), functional application (here, the application of $(\lambda \boldsymbol{x}.\, \boldsymbol{A})$ to $\boldsymbol{B}$), and the identification of co-extensional functions, respectively. In particular, β-conversion enables the substitution of all free occurrences of a variable in a formula with a suitably typed argument (here, the replacement of '$\boldsymbol{x}$' by '$\boldsymbol{B}$' in $\boldsymbol{A}$). η-conversion enables the replacement of $\lambda \boldsymbol{x}.\, \boldsymbol{A}(\boldsymbol{x})$ by $\boldsymbol{A}$ whenever $\boldsymbol{x}$ does not occur free in $\boldsymbol{A}$.[11]

[11] For an introduction to the lambda calculus, the reader is referred to (**Barendregt and Barendsen, 2000**), (**Hindley and Seldin, 2008**), and the interactive (**Barker, 2014**).

$$\frac{}{\Gamma, ⓘ \Rightarrow \Delta}\ ⓘ\mathsf{L} \qquad \frac{\Gamma, \boldsymbol{A} \Rightarrow \Delta, \boldsymbol{B}}{\Gamma \Rightarrow \Delta, (\boldsymbol{A} \doteRightarrow \boldsymbol{B})}\ \doteRightarrow \mathsf{R}$$

$$\frac{\Gamma, \boldsymbol{A}\{\boldsymbol{x} := \boldsymbol{B}\} \Rightarrow \Delta}{\Gamma, (\bigwedge \boldsymbol{x}.\, \boldsymbol{A}) \Rightarrow \Delta}\ \bigwedge \mathsf{L} \qquad \frac{\Gamma \Rightarrow \Delta, \boldsymbol{A}\{\boldsymbol{x} := \boldsymbol{c}\}}{\Gamma \Rightarrow \Delta, (\bigwedge \boldsymbol{x}.\, \boldsymbol{A})}\ \bigwedge \mathsf{R} \quad \text{where } \boldsymbol{c} \text{ is fresh}$$

$$\frac{\Gamma, \Delta, \boldsymbol{A} \doteq \boldsymbol{B} \Rightarrow \boldsymbol{C}\{\boldsymbol{x} := \boldsymbol{A}\}}{\Gamma, \Delta, \boldsymbol{A} \doteq \boldsymbol{B} \Rightarrow \boldsymbol{C}\{\boldsymbol{x} := \boldsymbol{B}\}}\ \doteq \mathsf{L} \qquad \frac{}{\Gamma \Rightarrow \Delta, \boldsymbol{A} \doteq \boldsymbol{A}}\ \doteq \mathsf{R}$$

$$\frac{\neg\Gamma \Rightarrow \Delta}{\neg\Delta \Rightarrow \Gamma}\ \neg\mathsf{L} \qquad \frac{\Gamma \Rightarrow \neg\Delta}{\Delta \Rightarrow \neg\Gamma}\ \neg\mathsf{R}$$

$$\frac{\Gamma, \boldsymbol{A}, \boldsymbol{B} \Rightarrow \Delta}{\Gamma, (\boldsymbol{A} \wedge \boldsymbol{B}) \Rightarrow \Delta}\ \wedge\mathsf{L} \qquad \frac{\Gamma \Rightarrow \Delta, \boldsymbol{A} \qquad \Gamma \Rightarrow \Delta, \boldsymbol{B}}{\Gamma \Rightarrow \Delta, (\boldsymbol{A} \wedge \boldsymbol{B})}\ \wedge\mathsf{R}$$

$$\frac{}{\Gamma, \boxdot \boldsymbol{A} \Rightarrow \Delta, \diamondsuit \boldsymbol{A}}\ \mathbf{D}$$

$$\frac{}{\Delta, \boxdot \boldsymbol{A} \Rightarrow \Gamma, \boxdot\boxdot \boldsymbol{A}}\ \mathbf{4} \qquad \frac{}{\Delta, \diamondsuit \boldsymbol{A} \Rightarrow \Gamma, \boxdot\diamondsuit \boldsymbol{A}}\ \mathbf{5}$$

$$\frac{\Gamma, \boldsymbol{A} \Rightarrow \Delta}{\Gamma, \boldsymbol{B} \Rightarrow \Delta}\ \lambda\mathsf{L} \quad \text{where } \boldsymbol{A} \doteq_{\beta\eta} \boldsymbol{B} \qquad \frac{\Gamma \Rightarrow \Delta, \boldsymbol{A}}{\Gamma \Rightarrow \Delta, \boldsymbol{B}}\ \lambda\mathsf{R} \quad \text{where } \boldsymbol{A} \doteq_{\beta\eta} \boldsymbol{B}$$

$$\frac{}{\big(\bigwedge \boldsymbol{x}.\, \boldsymbol{A}(\boldsymbol{x}) \leftrightarrows \boldsymbol{B}(\boldsymbol{x})\big) \Rightarrow \boldsymbol{A} \doteq \boldsymbol{B}}\ \mathsf{ext} \quad \text{if } \boldsymbol{x} \text{ is not free in } \boldsymbol{A} \text{ or } \boldsymbol{B}$$

TABLE 2.2. Logical rules for TY_0.

By the rule of α-conversion, the term $\lambda \boldsymbol{z} \lambda \boldsymbol{u}.\boldsymbol{find}(\boldsymbol{z}, \boldsymbol{u})$ (where $\boldsymbol{find}$ is a type-$(o\,o; o)$ constant, and where $\boldsymbol{z}$ and $\boldsymbol{u}$ are basic-type variables) will denote the same function as the term $\lambda \boldsymbol{y}_o \lambda \boldsymbol{x}_o.\boldsymbol{find}(\boldsymbol{y}, \boldsymbol{x})$. By β-conversion, the result, $\lambda \boldsymbol{x}_o.\boldsymbol{find}(\boldsymbol{bill}, \boldsymbol{x})$, of applying the predicate $\lambda \boldsymbol{y}_o \lambda \boldsymbol{x}_o.\boldsymbol{find}(\boldsymbol{y}, \boldsymbol{x})$ to the type-o constant $\boldsymbol{bill}$ will be equivalent to the unreduced term $\lambda \boldsymbol{y}_o \lambda \boldsymbol{x}_o.\boldsymbol{find}(\boldsymbol{y}, \boldsymbol{x})(\boldsymbol{bill})$.[12] The rule of η-conversion justifies our replacement of the term $\boldsymbol{find}$ by $\lambda \boldsymbol{y}_o \lambda \boldsymbol{x}_o.\boldsymbol{find}(\boldsymbol{y}, \boldsymbol{x})$.

Our translation of logical PTQ forms into TY_0 terms (in Chapter 3) will employ a large number of β-conversions. In that chapter, we will often use α-conversion to avoid the obtaining of non-equivalent TY_0 terms by 'variable collision'. For example, to avoid obtaining the term $\lambda \boldsymbol{x}.\boldsymbol{find}(\boldsymbol{x}, \boldsymbol{x})$ from the application of the predicate $\lambda \boldsymbol{y} \lambda \boldsymbol{x}.\boldsymbol{find}(\boldsymbol{y}, \boldsymbol{x})$ to the argument $\boldsymbol{x}$ (s.t. the lambda operator now wrongfully binds *two* variables), we rename the bound variable $\boldsymbol{x}$ in $\lambda \boldsymbol{y} \lambda \boldsymbol{x}.\boldsymbol{find}(\boldsymbol{y}, \boldsymbol{x})$ as '$\boldsymbol{z}$'. Subsequently, we will not use the TY_0 term $\lambda \boldsymbol{y} \lambda \boldsymbol{x}.\boldsymbol{find}(\boldsymbol{y}, \boldsymbol{x})$, but its alphabetic variant $\lambda \boldsymbol{y} \lambda \boldsymbol{z}.\boldsymbol{find}(\boldsymbol{y}, \boldsymbol{z})$. The application of this variant to the argument $\boldsymbol{x}$ then yields the 'correct' term $\lambda \boldsymbol{z}.\boldsymbol{find}(\boldsymbol{x}, \boldsymbol{z})$.

In virtue of the above and of Notation 2.1.1, the sequent rules for $\textcircled{\scriptsize T}$ and $\not=$, and for $\forall$, $\dot{\rightarrow}$, $\dot{\leftrightarrow}$, and $\bigvee$ are easily derivable (cf. Appendix C.1). The De Morgan laws (DM_1, DM_2), the rules of Double Negation (DNI, DNE), and the rule of Contraposition (CP) are also derivable. These rules are listed in Table 2.3:

$$\frac{\Gamma \Rightarrow \Delta, (\neg \boldsymbol{A} \vee \neg \boldsymbol{B})}{\Gamma \Rightarrow \Delta, \neg(\boldsymbol{A} \wedge \boldsymbol{B})}\ \mathsf{DM}_1 \qquad \frac{\Gamma \Rightarrow \Delta, (\neg \boldsymbol{A} \wedge \neg \boldsymbol{B})}{\Gamma \Rightarrow \Delta, \neg(\boldsymbol{A} \vee \boldsymbol{B})}\ \mathsf{DM}_2$$

$$\frac{}{\boldsymbol{A} \Rightarrow \neg\neg \boldsymbol{A}}\ \mathsf{DNI} \qquad \frac{}{\neg\neg \boldsymbol{A} \Rightarrow \boldsymbol{A}}\ \mathsf{DNE}$$

$$\frac{\Gamma \Rightarrow \Delta}{\neg\Delta \Rightarrow \neg\Gamma}\ \mathsf{CP}$$

TABLE 2.3. Some derived rules for TY_0.

We close this section by identifying some meta-mathematical properties of the logic TY_0. These properties include the soundness of the presented sequent calculus. This property is stated in Theorem 2.3:

THEOREM 2.3 (Soundness). *For all sets* Γ, Δ *of* TY_0 *terms, if* $\Gamma \vdash_{\mathrm{TY}_0} \Delta$, *then* $\Gamma \models_g \Delta$.

PROOF. The proof is included in Appendix C.2.

We prove the generalized completeness of the logic TY_0 – together with its compactness and the Löwenheim-Skolem property – via a Model Existence theorem,

[12] To ease readability, Chapter 3 will often place arguments in square brackets. The TY_0 term $\lambda \boldsymbol{y} \lambda \boldsymbol{x}.\boldsymbol{find}(\boldsymbol{y}, \boldsymbol{x})(\boldsymbol{bill})$ will then appear as '$\lambda \boldsymbol{y} \lambda \boldsymbol{x}.\boldsymbol{find}(\boldsymbol{y}, \boldsymbol{x})[\boldsymbol{bill}]$'.

cf. (**Smullyan, 1995**). This theorem takes the following form:

THEOREM 2.4 (Model Existence). *Let L and $\mathcal{C}$ be* TY_0 *languages s.t. $L \cap \mathcal{C} = \emptyset$, where every set $\mathcal{C}_\alpha$ is countably infinite. Assume that $\mathfrak{P}$ is a sound provability property w.r.t. $L \cup \mathcal{C}$, and that Π is a sequent in L. If $\Pi \notin \mathfrak{P}$, then Π is refutable by a countable* TY_0 *model.*

PROOF. The proof can be found in Appendix C.2.

From Theorem 2.4, we can show that the following holds, where Π is as above, and where Σ ranges over sequents in $L \cup \mathcal{C}$:

COROLLARY 2.1 (Generalized Compactness). *For all* TY_0 *sequents* Π *and* TY_0 *models M, if, for all finite $\Pi_0 \subseteq \Pi$, $M \models_g \Pi_0$, then $M \models_g \Pi$.*

PROOF. The set $\{\Sigma \mid M \models_g \Sigma_0 \text{ for some finite } \Sigma_0 \subseteq \Sigma\}$ is a sound provability property. □

COROLLARY 2.2 (Generalized Löwenheim-Skolem). *For all* TY_0 *sequents* Π*, if $M \not\models_g \Pi$, then Π is refutable by a countable* TY_0 *model.*

PROOF. The set $\{\Sigma \mid M \models_g \Sigma\}$ is a sound provability property. □

COROLLARY 2.3 (Generalized Completeness). *For all finite sets Γ, Δ of* TY_0 *terms, if $\Gamma \models_g \Delta$, then $\Gamma \vdash_{\mathrm{TY}_0} \Delta$.*

PROOF. The set $\{\Sigma \mid \Sigma \text{ is } \mathrm{TY}_0\text{-provable}\}$ is a sound provability property. □

We close the chapter with an attempt to supply truth-conditions for basic-type TY_0 terms.

2.5. Ways to Truth

In Section 2.3, we have observed the difficulty of providing a truth-definition for basic-type terms of the logic TY_0. We have attributed this difficulty to the unavailability of a TY_0 type t for truth-values (at the level of the object theory).

The present section identifies three strategies for the truth-evaluation of type-o terms.[13] These strategies include the specification of a homomorphism from entities to (functions from indices to) truth-values, the representation of indices via complex objects in the metatheory, and the identification of entities with their evaluating indices. The strategies are derived from the evaluation of proposition-denoting formulas in Thomason's (**1980**) *Intentional Logic* [*sic*], in Pollard's (**2008**) *constructed worlds theory*, and in Fine's (**1982**) theory of *worlds as facts*. We discuss the three strategies in turn. In this discussion, we will first specify the original formulation of these strategies. We will then adapt them for our purposes. We close by assessing the usability of the resulting truth-definitions.

[13]This section was prompted by discussions with Daisuke Bekki, Markus Werning, and Dietmar Zaefferer.

2.5.1. Truth-Homomorphisms. Intentional Logic (**Thomason, 1980**), cf. (**Chierchia and Turner, 1988**; **Fox and Lappin, 2004**; **Muskens, 2005**), is a theory of formal semantics which replaces Montagovian propositions (type $\langle s, t\rangle$) by semantically primitive propositions (type p). To enable the truth-evaluation of proposition-denoting formulas, Thomason retains the truth-value type t. A homomorphism from primitive propositions to truth-values determines a formula's truth or falsity at the current index. By supplementing a basic type for indices, s, and by adapting the homomorphism to a function from primitive propositions to *Montagovian* propositions (type $\langle s, t\rangle$), Muskens (**2005**) generalizes Thomason's homomorphism to other indices. As a result, formulas in Intentional Logic have their usual truth- and falsity-conditions.

Since Thomason's primitive propositions compare to entities in our TY_0-based single-type semantics, Intentional Logic suggests a strategy for the truth-evaluation of basic-type TY_0 terms. This strategy involves the adoption of the additional basic types s and t in the TY_0 object theory, and the specification of a function from entities to Montagovian propositions (Strategy (1a)). However, since the resulting semantics assumes *three* basic types o, s, and t, it violates Partee's conjecture from Proposition 1.2.

The confinement of the types s and t to the TY_0 metatheory, and the delegation of truth-evaluation to models of this theory (Strategy (1b)), solves the above problem. The resulting truth-definition then proceeds analogously to the definition of TY_0 entailment from Section 2.4. A variant of this strategy is adopted in (**Fox and Lappin, 2005**, Ch.5), cf. (**Fox and Lappin, 2004**; **Turner, 1992**). However, it remains questionable whether the assumption of a homomorphism between elements of the object theory and elements of the metatheory is methodologically admissible. The extension of the metatheoretical type system by the type o (Strategy (1c)) also solves this new problem. However, this extension would require the introduction of a metatheory which is even more complex (and more richly typed) than the proposed metatheory from Section 2.3.

	(1)			(2)		(3)
	(a)	(b)	(c)	(a)	(b)	
Object th'y types	o, s, t	o	o	o, t	o	o
Metatheory types	o, s, t	s, t	o, s, t	o, t	o, t	o, t

TABLE 2.4. Strategies for obtaining TY_0 truth.

The different 'Thomasonian' strategies for the evaluation of TY_0 truth are summarized in the three leftmost columns of Table 2.4 (previous page).

2.5.2. Constructed Worlds. The semantics of constructed worlds from (**Pollard, 2008**), cf. (**Fox and Lappin, 2005**; **Pollard, 2005**), improves upon Strategy (1) by dispensing with the need for an index type s, and by waiving the requirement of a homomorphism between primitive and Montagovian objects. To achieve this, Pollard (**2008**) replaces indices by *constructed* (*possible*) *worlds*. The latter are ultrafilters in the set of propositions, which are associated with a subset of the set of objects of the type $\langle p, t\rangle$, cf. (**Veltman, 1981**; **Scott, 1982**; **Landman, 1985**). The type for constructed worlds inverses the familiar relation between propositions and worlds. Thus, in constructed worlds semantics, a proposition-denoting formula φ is true at a constructed world w if $\varphi \in w$ (not if $w \in \varphi$, as is the case in Montague semantics).

The replacement of primitive propositions (type p) by primitive entities (type o) adapts Pollard's truth-definition to our single-type semantics. Naturally, the compliance of our revised single-type semantics with Partee's conjecture prevents the introduction of the truth-value type t (and thus, of the type for constructed worlds $(o; t)$) into the TY_0 object theory (Strategy (2a)). The adoption of the type t into the type system of the metatheory (Strategy (2b)) complies with Partee's conjecture and with our description of the TY_0 metatheory from Section 2.3. However, the proposed semantics obfuscates the truth-contribution of sentential constituents (cf. Ch. 1.1.1): the semantic values of basic non-sentential expressions (e.g. of (in-)transitive verbs) can only be indirectly identified with their contribution to the truth-conditions of the sentences in which they occur. We will see in Part III that there exist other single-type semantics which do not rely on the coding of worlds as 'new' metatheoretical objects.

The two middle columns of Table 2.4 summarize 'constructed worlds' strategies for the evaluation of TY_0 truth. We finish our presentation of different truth-evaluative strategies with an alternative to Strategy (2).

2.5.3. Worlds as Entities. Fine's theory of 'worlds as facts' (**Fine, 1982**), cf. (**Barwise and Perry, 1983**; **Kratzer, 1989**; **2011**), constitutes a variant of constructed worlds semantics which represents indices without the use of the truth-value type t. In particular, Fine regards possible worlds as "very big facts" (**Fine, 1982**, p. 43) (in our theory, semantically very 'rich' entities), which are obtained by combining the information of all true facts (i.e. a subset of our possible entities) at a constructed world. As a result, Fine's theory enables the definition of truth at a type-o world via inclusion *in* the world. Thus, a type-o term $\boldsymbol{A}$ is true at a world w if $w \subseteq \boldsymbol{A}$. The term $\boldsymbol{A}$ is false at w if w contains the De Morgan complement of the entity denoted by $\boldsymbol{A}$ (i.e. if $w \subseteq {}^{\frown}\boldsymbol{A}$).

Because of its particular definition of possible worlds, Fine's theory obviates the need to adopt the type t in the TY_0 object theory. However, the definition of truth in terms of *inclusion of semantic content* again requires the adoption of the type t in the metatheory (Strategy (3)). As a result, this strategy will share the merits and limitations of the constructed worlds approach (2b) from Section 2.5.2.

The above considerations have shown the high methodological cost of a truth-evaluation for TY_0 terms: While Strategy (1) assumes a homomorphism between entities and propositions, Strategies (2) and (3) require the additional coding of worlds. But this contradicts the alleged greater *simplicity* of single-type semantics (cf. Ch. 1.1.3, 1.4.1). We will see in Part III that the interpretation of the type o as a particular Montague type obviates the need for any of these additional complications.

2.6. Summary

This chapter has introduced the 'pure' single-type logic TY_0. This logic is a variant of Henkin's Theory of Propositional Types, whose basic type, o (for *entities*), remains unanalyzed. Types of this logic are obtained from o via a generalized variant of the type-forming rule **ST**. We have specified terms, frames, and models of the logic TY_0, have defined the relation of entailment on TY_0 terms, and have provided the logic with a sound and complete Gentzen-style sequent calculus.

Its single-typing lends the logic TY_0 a number of special properties. In particular, since the truth-value type t is not available in the TY_0 type system, the logic TY_0 requires single-type proxies for many familiar connectives and quantifiers, disables an easy truth-definition, and constrains the proxies' semantic behavior through the use of meta-level axioms. The notions of TY_0 entailment, equivalence, and provability are all defined in the metatheory. As a result, the availability of these notions places non-trivial conditions on Partee's conjecture.

We will see in the next chapter that the particular properties of the logic TY_0 render its models inadequate as a single-type semantics for the PTQ fragment.

CHAPTER 3

TY_0-Based Single-Type Semantics

The present chapter provides a TY_0-based single-type semantics for Montague's PTQ fragment. This semantics is a theory of meaning for the subset of English from (**Montague, 1973**), where all meaning is represented by constructions out of primitive *entities* (type o). Since the type o is neutral between Montague's types e and $\langle s,t\rangle$, proper names, sentences, and complement phrases all receive an interpretation in this type (Prop. 1.2). As a result, the semantics accommodates NP/CP complement-neutral verbs (cf. Ch. 1.2.1, (1)–(3)), NP/CP coordinations (cf. (4)–(6)), and CP equatives (cf. (7)) and accounts for the sentence-type interpretation of proper names (cf. (9), (10)).

The previous chapter has developed a general class of languages and models for the single-type logic TY_0. To enable the compositional interpretation of logical PTQ forms into TY_0 objects (along the lines of Ch. 1.1.1, Fig. 1.1), we further need to identify a designated TY_0 language $\mathcal{L}$, frame $\mathcal{F}$, and interpretation function $\mathcal{I}_\mathcal{F}$ (cf. step 2), provide a theory of syntax for the PTQ fragment (cf. step 1), and specify the translation of logical PTQ forms into TY_0 terms (cf. step 3).

The present chapter takes on this task. Its organization is as follows: Section 3.1 identifies the lexicon and syntax of the PTQ fragment. To enable the interpretation of PTQ expressions into objects of the logic TY_0, Section 3.2 introduces a set of translation rules, which define the translation of logical PTQ forms into TY_0 terms. The translation of all PTQ words will then enable an inductive translation of all logical PTQ forms. In virtue of their translation into terms of the logic TY_0, PTQ forms will display equivalence and entailment relations to other structures. Section 3.3 defines the notion of logical consequence on expressions of the fragment, and points out some striking semantic properties. The chapter closes by observing the difficulty of explaining syntactic well-formedness in a TY_0-based single-type semantics (in Sect. 3.4).

3.1. A Syntax for the PTQ Fragment

We start by presenting a rudimentary framework of PTQ syntax which gives the source of our TY_0 translations. To keep the framework as simple as possible, we allow the syntax of our fragment to overgenerate considerably. However, since we

are only interested in providing a precise syntactic terminology, overgeneration does not constitute a particular problem. For the present purposes, we assume a reduced variant of the Government and Binding theory from (**Chomsky, 1981**), cf. (**Muskens, 1996**; **Roelofsen, 2008**). The prominence of generative grammar in syntactic theory makes this adoption representative.[1]

Following Chomsky, we assume four levels (or 'components') of syntactic representation, called *Deep Structure* (D-*Structure*, or DS), *Surface Structure* (S-*Structure*, or SS), *Logical Form* (LF), and *Phonological Form* (PF). The first three components are defined as sets of trees or labelled bracketings, which are connected by movement rules. We assume that S-Structure is definable from D-Structure via wh-Movement, and that Logical Form is definable from S-Structure by Quantifier Raising. Phonological Form is also definable from S-Structure. The relation between the different syntactic components is depicted in Figure 3.1:

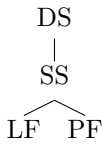

FIGURE 3.1. The relation between DS, SS, LF, and PF.

The trees of our syntax' DS-component can be generated by the phrase structure rules (PS 1–11) from Table 3.2 (next but one page) and the lexical insertion rules (LI 1–12) from Table 3.1 (next page). To enable the interpretation of the sentences (1), (2), (7a), and (9) from Chapter 1.2.1, we extend the PTQ fragment by the names Pat, Sherlock, Moriarty, and Barbara Partee, the common nouns problem and room, the intransitive verbs wait and arrive, the transitive verbs remember, fear, destroy, hate, and enter, and the preposition for. For technical reasons (cf. Ch. 4.2), we further extend the fragment with the intransitive verb exist and the sentence adverb possibly. In Table 3.1, non-PTQ words are written in grey font.

Examples of labelled bracketings (or trees) which are elements of D-Structure are included below:

(3.1.1) $[_{S}[_{NP}[_{DET}\mathsf{a}][_{N}\mathsf{man}]][_{VP}[_{IV}\mathsf{walks}]]]$

(3.1.2) $[_{S}[_{NP}\mathsf{Mary}][_{VP}[_{SCV}\mathsf{believes}][_{CP}[_{C}\mathsf{that}][_{S}[_{NP}\mathsf{John}][_{VP}[_{TV}\mathsf{finds}][_{NP}[\mathsf{a}][_{N}\mathsf{unicorn}]]]]]]]$

(3.1.3) $[_{NP}[_{DET}\mathsf{a}][_{N}\mathsf{woman}]][_{S}[_{NP}[_{DET}\mathsf{every}][_{N}\mathsf{man}]][_{VP}[_{TV}\mathsf{loves}][_{NP}\mathsf{whom}]]]$

[1]Anna Szabolcsi has suggested the better suitability of Distributed Morphology as a syntax for single-type semantics. Distributed Morphology is introduced in (**Harley, forthcoming**) and in (**Harley and Noyer, 2003**). However, to avoid the introduction of a new syntactic framework *in addition to* a new semantics, we here refrain from adopting this option.

Label	Rule			Traditional Name
(LI 1)	DET	$\longrightarrow$	every, the, a	Determiner
(LI 2)	NP	$\longrightarrow$	John, Mary, Bill, ninety, he_n, she_n, it_n, Pat, Sherlock, Moriarty, Barbara Partee	Noun Phrase
(LI 3)	NP	$\longrightarrow$	who, whom, which	Noun Phrase
(LI 4)	N	$\longrightarrow$	man, woman, park, fish, pen, unicorn, price, temperature, problem, room	Common Noun
(LI 5)	IV	$\longrightarrow$	run, walk, talk, rise, change, wait, arrive, exist	Intransitive Verb
(LI 6)	TV	$\longrightarrow$	find, lose, eat, love, date, be, seek, conceive, fear, remember, destroy, hate, enter	Transitive Verb
(LI 7)	SAV	$\longrightarrow$	necessarily, possibly	Sentence Adverb
(LI 8)	SCV	$\longrightarrow$	believe, assert	Sentence-Complement Verb
(LI 9)	C	$\longrightarrow$	that	Complementizer
(LI 10)	ADV	$\longrightarrow$	rapidly, slowly, voluntarily, allegedly	Verb Phrase Adverb
(LI 11)	P	$\longrightarrow$	in, about, for	Preposition
(LI 12)	ICV	$\longrightarrow$	try to, wish to	Infinitive-Complement Verb

Table 3.1. Lexical insertion rules.

Label	Rule			Traditional Name
(PS 1)	S	$\longrightarrow$	NP VP	Sentence
(PS 2)	S	$\longrightarrow$	SAV S	Sentence
(PS 3)	NP	$\longrightarrow$	DET N	Noun Phrase
(PS 4)	CP	$\longrightarrow$	C S	Complement Phrase
(PS 5)	VP	$\longrightarrow$	IV	Verb Phrase
(PS 6)	VP	$\longrightarrow$	TV NP	Verb Phrase
(PS 7)	VP	$\longrightarrow$	SCV CP	Verb Phrase
(PS 8)	VP	$\longrightarrow$	ADV IV	Verb Phrase
(PS 9)	VP	$\longrightarrow$	VP PP	Verb Phrase
(PS 10)	VP	$\longrightarrow$	IV ICV	Verb Phrase
(PS 11)	PP	$\longrightarrow$	P NP	Prepositional Phrase

Table 3.2. Phrase structure rules for the generation of Deep Structures.

We next turn to the definition of S-Structure. Trees of the latter are understood as acceptable forms of English which will be interpretable in our semantics. S-structures are connected to D-structures by a rudimentary version of wh-Movement. The latter is an operation whereby a wh-element (e.g. the relative pronoun $[_{\text{NP}}\mathsf{who}]$) is moved from the position where it was generated to some S-node at a higher position in the tree. The moved element receives a binder index n, which is adjoined to it in superscript (e.g. $[_{\text{NP}}\mathsf{who}]^3$), and leaves a trace t_n (here, t_3), with which it becomes coindexed. Wh-Movement is formally defined as follows:

(3.1.4) $[_{\text{S}}X\ [_{\text{NP}}\mathsf{wh}]\ Y] \quad \Rightarrow \quad [_{\text{S}}[_{\text{NP}}\mathsf{wh}]^n\ [_{\text{S}}X\ t_n\ Y]]$ (wh-Movement)

The tree in (3.1.5) follows from (3.1.3) by wh-Movement:

(3.1.5) $[_{\text{NP}}[_{\text{DET}}\mathsf{a}][_{\text{N}}\mathsf{woman}]][_{\text{S}}[_{\text{NP}}\mathsf{whom}]^1[_{\text{S}}[_{\text{NP}}[_{\text{DET}}\mathsf{every}][_{\text{N}}\mathsf{man}]][_{\text{VP}}[_{\text{TV}}\mathsf{loves}]\ t_1]]]$

As a result, S-Structure is definable as the smallest set of trees which contains D-Structure and which is closed under wh-Movement (s.t. DS $\subseteq$ SS).

Since phonology exceeds the scope of this dissertation, we do not attempt to give a serious definition of Phonological Form. For the present purposes, we simply assume that phonological forms are obtained from surface structures through the deletion of all brackets, indices, and traces. Thus, (3.1.6) is the phonological form associated with (3.1.5):

(3.1.6) A woman whom every man loves

We finish our discussion of the different components of syntactic representation with a definition of Logical Form. The latter is the component of grammar which is interpreted by our TY$_0$ semantics. The motivation for having a Logical Form-component in syntax is entirely semantic in nature: Traditional grammar only of-

fers a single syntactic structure for sentences involving different quantifiers or intensional operators. Thus, on the level of S-Structure, the sentence Every man loves a woman only has a single representation, i.e. (3.1.7), in which the subject (every man) *c-commands* (or takes scope over) the object (a woman). This representation matches the left-to-right order of the quantifiers.

(3.1.7) $[_S[_{NP}[_{DET}\mathsf{every}][_N\mathsf{man}]][_{VP}[_{TV}\mathsf{loves}][_{NP}[_{DET}\mathsf{a}][_N\mathsf{woman}]]]]$

Semantic ambiguity is an expected feature of S-Structure. However, we want to account for the different readings of (3.1.7), paraphrased in (3.1.8) and (3.1.9):

(3.1.8) Every man loves some (poss. different) woman (from every other man).

(3.1.9) There is a woman whom every man loves.

From (3.1.7), the reading in (3.1.9), which 'inverts' the scope of the subject and object (s.t. a woman c-commands every man), is obtained through Quantifier Raising. The latter is an operation due to (**May, 1977**), cf. (**Lakoff, 1970**; **May, 1985**; **Heim and Kratzer, 1998**; **Reinhart, 2006**), whereby a noun phrase is adjoined to a sentence:

(3.1.10) $[_S X\ [_{NP} Z]\ Y] \quad \Rightarrow \quad [_S[_{NP} Z]^n\ [_S X\ t_n\ Y]]$ (Quantifier Raising)

The tree in (3.1.11) follows from the S-structure in (3.1.7) by Quantifier Raising:

(3.1.11) $[_S[_{NP}[_{DET}\mathsf{a}][_N\mathsf{woman}]]^2[_S[_{NP}[_{DET}\mathsf{every}][_N\mathsf{man}]][_{VP}[_{TV}\mathsf{loves}]\ t_2]]]$

We define Logical Form as the smallest set of trees which contains S-Structure and which is closed under Quantifier Raising (s.t. SS $\subseteq$ LF). Thus, Quantifier Raising (like wh-Movement) is an optional rule. However, in contrast to (3.1.3), structures like (3.1.7) and (3.1.11) receive a semantic interpretation.

3.2. A TY_0 Semantics for the PTQ Fragment

We next specify a set of rules which govern the translation of the LF constitutents of Montague's PTQ fragment into terms of the logic TY_0.

3.2.1. From LF Constituents to TY_0 Terms. For reasons of simplicity, we opt for a *type-driven* translation of logical forms, as proposed by Klein and Sag (**1985**). The use of such a general translation procedure will spare us the stipulation of a separate semantic rule for each syntactic rule along the lines of (**Montague, 1973**). In particular, the specification of the TY_0 translations of logical PTQ forms proceeds in two steps, by first defining the translation of lexical elements (or *words*) of the fragment, and then defining the translation of non-lexical elements compositionally from the translation of their constituents.

Below, we let X, Y and $\boldsymbol{A}, \boldsymbol{B}$ be logical forms and TY_0 terms of the appropriate category, respectively type.

DEFINITION 3.2.1 (Type-driven translation). The *translation* relation $\rightsquigarrow$ is the smallest relation between trees and TY_0 terms which conforms to the rules (T0) to (T5):

(T0) If X is a word and $\boldsymbol{A}$ its translation, then $X \rightsquigarrow \boldsymbol{A}$. (Base Rule)

(T1) If $X \rightsquigarrow \boldsymbol{A}$, then $[X] \rightsquigarrow \boldsymbol{A}$. (Copying)

(T2) If $X \rightsquigarrow \boldsymbol{A}$ and $Y \rightsquigarrow \boldsymbol{B}$, then, if $\boldsymbol{A}(\boldsymbol{B})$ is well-formed, $[XY] \rightsquigarrow \boldsymbol{A}(\boldsymbol{B})$; otherwise, $[XY] \rightsquigarrow \boldsymbol{B}(\boldsymbol{A})$. (Application)

(T3) If $X \rightsquigarrow \boldsymbol{A}$ and $Y \rightsquigarrow \boldsymbol{B}$, then, if $\boldsymbol{A}(\lambda v_n.\boldsymbol{B})$ is well-formed, $[X^n Y] \rightsquigarrow \boldsymbol{A}(\lambda v_n.\boldsymbol{B})$. (Quantifying In)

(T4) If $X \rightsquigarrow \boldsymbol{A}$ and $\boldsymbol{A}$ is reducible to $\boldsymbol{B}$, then $X \rightsquigarrow \boldsymbol{B}$. (Reduction)

(T5) If $X \rightsquigarrow \boldsymbol{A}$ and $\boldsymbol{A}$ shifts to $\boldsymbol{B}$, then $X \rightsquigarrow \boldsymbol{B}$. (Type-Shifting)

Rule (T0) constitutes the 'base rule' of type-driven translation. This rule specifies the translation of lexical elements of the PTQ fragment, such that their instances are sent to TY_0 terms. Rules (T1) to (T5) specify how the translation of complex logical forms depends on the translation of their daughter nodes. In particular, rules (T1) and (T2) specify the translation of mothers of a single, and of two daughters. While mothers of a single daughter simply inherit their daughter's translation, mothers of two daughters are translated as the result of the functional application of the translation of one to the translation of the other daughter. Rule (T3) handles quantifying-in. Rules (T4) and (T5) enable the simplification and type-shifting of translations. The former employs the usual rules, α, β, η, of lambda conversion. Thus, we say that a TY_0 term $\boldsymbol{B}$ *follows by reduction* from a term $\boldsymbol{A}$ if we can obtain $\boldsymbol{B}$ from $\boldsymbol{A}$ by a finite number of lambda conversions. Some type-shifting rules will be introduced below. In all of the above clauses, translation is accepted modulo logical equivalence (s.t. $X \rightsquigarrow \boldsymbol{B}$ if $X \rightsquigarrow \boldsymbol{A}$ and $\boldsymbol{A} = \boldsymbol{B}$).

Notably, the relation $\rightsquigarrow$ is neither functional nor total. As a result of the former, a single logical form may be ambiguous between several non-equivalent TY_0 translations (cf. (T5)). As a result of the latter, some forms may lack a TY_0 translation (s.t. they are uninterpretable). We exclude uninterpretable forms from our further considerations.

3.2.2. Fixing $\mathcal{L}$ and $\mathcal{F}$. To enable the specific translation of the Logical Form-constituents of Montague's PTQ fragment, we first define a particular TY_0 language $\mathcal{L}$ and frame $\mathcal{F}$. The members of $\mathcal{L}$ are specified in Table 3.3 (next page). Our conventions for the use of TY_0 variables are introduced in Table 3.4. Since some of the designated TY_0 constants from Definition 2.1.2 and Notation 2.1.1 (e.g. the constants $\doteq$, $\wedge$, $\bigvee$, and $\boxdot$) will figure in our translation of logical PTQ forms, we assume their membership in $\mathcal{L}$.

To give a general interpretation of the Logical Form constitutents of Mon-

CONSTANT	TY_0 TYPE
$\ulcorner$	$((\alpha_1 \ldots \alpha_n; o)\, \alpha_1 \ldots \alpha_n; o)$
$\wedge, \vee$	$((\alpha_1 \ldots \alpha_n; o)\, (\alpha_1 \ldots \alpha_n; o)\, \alpha_1 \ldots \alpha_n; o)$
$\bigwedge, \bigvee$	$(\alpha\, o; o)$
$\dot{\Rightarrow}, \doteq, \dot{\neq}, \dot{\rightarrow}, \dot{\leftrightarrow}$	$(\alpha\, \alpha; o)$
$\top, \bot$, ***john***, ***mary***, ***bill***, ***ninety***, ***sherlock***, ***moriarty***, ***pat***, ***partee***, ***w***	o
$\boxdot, \diamond$, ***man***, ***woman***, ***park***, ***fish***, ***pen***, ***unicorn***, ***room***, ***problem***	$(o; o)$
run, ***walk***, ***talk***, ***wait***, ***arrive***, ***E***	$(o; o)$
find, ***lose***, ***eat***, ***love***, ***date***, ***remember***, ***fear***, ***destroy***, ***hate***, ***enter***, ***believe***, ***assert***	$(o\, o; o)$
temp, ***price***, ***rise***, ***change***	$((o; o); o)$
rapidly, ***slowly***, ***voluntary***, ***allegedly***, ***try***, ***wish***	$((o; o)\, o; o)$
in, ***for***	$(o\, (o; o)\, o; o)$
seek, ***conceive***	$(((o; o); o)\, o; o)$
about	$(((o; o); o)\, (o; o)\, o; o)$

TABLE 3.3. $\mathcal{L}$ constants.

VARIABLE	TY_0 TYPE	OBJECT
$\boldsymbol{x}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n, \boldsymbol{y}, \boldsymbol{z}$	o	entity
$\boldsymbol{p}, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_n, \boldsymbol{q}, \boldsymbol{r}$	o	entity
$\boldsymbol{P}, \boldsymbol{P}_1, \ldots, \boldsymbol{P}_n$	$(o; o)$	1^{st}-order property of entities
$\boldsymbol{Q}, \boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_n$	$((o; o); o)$	2^{nd}-order property of entities
$\boldsymbol{L}, \boldsymbol{L}_1, \ldots, \boldsymbol{L}_n$	$(((o; o); o)\, o; o)$	

TABLE 3.4. TY_0 variables.

tague's PTQ fragment, we let the TY_0 frame $\mathcal{F}$ be very large, such that it contains possible semantic values of all elements in $\mathcal{L}$. The designated interpretation function $\mathcal{I}_\mathcal{F}$ sends all constants of $\mathcal{L}$ to their associated semantic values in $\mathcal{F}$. In particular, we let $\mathcal{I}_\mathcal{F}$ be the most general function which respects the way in which different content words are conventionally related.[2] The specific role of the interpretation function $\mathcal{I}_\mathcal{F}$ will be discussed in Part III (cf. Ch. 7.2.2, 8.2.2).

3.2.3. Translations of Atomic Logical Forms. We next translate the lexical elements of the fragment.

DEFINITION 3.2.2 (Basic TY_0 translations). The translation rule (T0) associates the lexical elements from Table 3.1 with the following TY_0 terms:

[2]Thus, $\mathcal{I}_\mathcal{F}$ is such that $\mathcal{I}_\mathcal{F}(\lambda \boldsymbol{x}.\, \boldsymbol{bill} \doteq \boldsymbol{x}) \subseteq \mathcal{I}_\mathcal{F}(\boldsymbol{man})$, where $\lambda \boldsymbol{x}.\, \boldsymbol{bill} \doteq \boldsymbol{x}$ and $\boldsymbol{man}$ are the TY_0 translations of the verb phrase be Bill and the common noun man, respectively (cf. Def. 3.2.2).

John	$\rightsquigarrow$	$\boldsymbol{john}$;	Mary	$\rightsquigarrow$	$\boldsymbol{mary}$;
Bill	$\rightsquigarrow$	$\boldsymbol{bill}$;	ninety	$\rightsquigarrow$	$\boldsymbol{ninety}$;
Pat	$\rightsquigarrow$	$\boldsymbol{pat}$;	Barbara Partee	$\rightsquigarrow$	$\boldsymbol{partee}$;
Sherlock	$\rightsquigarrow$	$\boldsymbol{sherlock}$;	Moriarty	$\rightsquigarrow$	$\boldsymbol{moriarty}$;
man	$\rightsquigarrow$	$\boldsymbol{man}$;	who(m)/which	$\rightsquigarrow$	$\lambda \boldsymbol{P}.\boldsymbol{P}$;
fish	$\rightsquigarrow$	$\boldsymbol{fish}$;	woman	$\rightsquigarrow$	$\boldsymbol{woman}$;
park	$\rightsquigarrow$	$\boldsymbol{park}$;	unicorn	$\rightsquigarrow$	$\boldsymbol{unicorn}$;
pen	$\rightsquigarrow$	$\boldsymbol{pen}$;	temperature	$\rightsquigarrow$	$\boldsymbol{temp}$;
price	$\rightsquigarrow$	$\boldsymbol{price}$;	fears	$\rightsquigarrow$	$\lambda \boldsymbol{Q}\lambda \boldsymbol{x}.\boldsymbol{Q}(\lambda \boldsymbol{y}.\boldsymbol{fear}(\boldsymbol{y},\boldsymbol{x}))$;
problem	$\rightsquigarrow$	$\boldsymbol{problem}$;	hates	$\rightsquigarrow$	$\lambda \boldsymbol{Q}\lambda \boldsymbol{x}.\boldsymbol{Q}(\lambda \boldsymbol{y}.\boldsymbol{hate}(\boldsymbol{y},\boldsymbol{x}))$;
room	$\rightsquigarrow$	$\boldsymbol{room}$;	finds	$\rightsquigarrow$	$\lambda \boldsymbol{Q}\lambda \boldsymbol{x}.\boldsymbol{Q}(\lambda \boldsymbol{y}.\boldsymbol{find}(\boldsymbol{y},\boldsymbol{x}))$;
waits	$\rightsquigarrow$	$\boldsymbol{wait}$;	loses	$\rightsquigarrow$	$\lambda \boldsymbol{Q}\lambda \boldsymbol{x}.\boldsymbol{Q}(\lambda \boldsymbol{y}.\boldsymbol{lose}(\boldsymbol{y},\boldsymbol{x}))$;
arrives	$\rightsquigarrow$	$\boldsymbol{arrive}$;	eats	$\rightsquigarrow$	$\lambda \boldsymbol{Q}\lambda \boldsymbol{x}.\boldsymbol{Q}(\lambda \boldsymbol{y}.\boldsymbol{eat}(\boldsymbol{y},\boldsymbol{x}))$;
runs	$\rightsquigarrow$	$\boldsymbol{run}$;	loves	$\rightsquigarrow$	$\lambda \boldsymbol{Q}\lambda \boldsymbol{x}.\boldsymbol{Q}(\lambda \boldsymbol{y}.\boldsymbol{love}(\boldsymbol{y},\boldsymbol{x}))$;
walks	$\rightsquigarrow$	$\boldsymbol{walk}$;	dates	$\rightsquigarrow$	$\lambda \boldsymbol{Q}\lambda \boldsymbol{x}.\boldsymbol{Q}(\lambda \boldsymbol{y}.\boldsymbol{date}(\boldsymbol{y},\boldsymbol{x}))$;
talks	$\rightsquigarrow$	$\boldsymbol{talk}$;	destroys	$\rightsquigarrow$	$\lambda \boldsymbol{Q}\lambda \boldsymbol{x}.\boldsymbol{Q}(\lambda \boldsymbol{y}.\boldsymbol{destroy}(\boldsymbol{y},\boldsymbol{x}))$;
exists	$\rightsquigarrow$	$\boldsymbol{E}$;	remembers	$\rightsquigarrow$	$\lambda \boldsymbol{Q}\lambda \boldsymbol{x}.\boldsymbol{Q}(\lambda \boldsymbol{y}.\boldsymbol{remember}(\boldsymbol{y},\boldsymbol{x}))$;
rises	$\rightsquigarrow$	$\boldsymbol{rise}$;	enters	$\rightsquigarrow$	$\lambda \boldsymbol{Q}\lambda \boldsymbol{x}.\boldsymbol{Q}(\lambda \boldsymbol{y}.\boldsymbol{enter}(\boldsymbol{y},\boldsymbol{x}))$;
changes	$\rightsquigarrow$	$\boldsymbol{change}$;	is	$\rightsquigarrow$	$\lambda \boldsymbol{Q}\lambda \boldsymbol{x}.\boldsymbol{Q}(\lambda \boldsymbol{y}.\boldsymbol{x} \doteq \boldsymbol{y})$;
seeks	$\rightsquigarrow$	$\boldsymbol{seek}$;	believes	$\rightsquigarrow$	$\lambda \boldsymbol{p}\lambda \boldsymbol{Q}.\boldsymbol{Q}(\lambda \boldsymbol{x}.\boldsymbol{believe}(\boldsymbol{p},\boldsymbol{x}))$;
conceives	$\rightsquigarrow$	$\boldsymbol{conceive}$;	asserts	$\rightsquigarrow$	$\lambda \boldsymbol{p}\lambda \boldsymbol{Q}.\boldsymbol{Q}(\lambda \boldsymbol{x}.\boldsymbol{assert}(\boldsymbol{p},\boldsymbol{x})) \wedge \boldsymbol{p}$;
about	$\rightsquigarrow$	$\boldsymbol{about}$;	in	$\rightsquigarrow$	$\lambda \boldsymbol{Q}\lambda \boldsymbol{P}\lambda \boldsymbol{x}.\boldsymbol{Q}(\lambda \boldsymbol{y}.\boldsymbol{in}(\boldsymbol{y},\boldsymbol{P},\boldsymbol{x})$;
allegedly	$\rightsquigarrow$	$\boldsymbol{allegedly}$;	for	$\rightsquigarrow$	$\lambda \boldsymbol{Q}\lambda \boldsymbol{P}\lambda \boldsymbol{x}.\boldsymbol{Q}(\lambda \boldsymbol{y}.\boldsymbol{for}(\boldsymbol{y},\boldsymbol{P},\boldsymbol{x})$;
that	$\rightsquigarrow$	$\lambda \boldsymbol{p}.\boldsymbol{p}$;	slowly	$\rightsquigarrow$	$\lambda \boldsymbol{P}\lambda \boldsymbol{x}.\boldsymbol{slowly}(\boldsymbol{P},\boldsymbol{x}) \wedge \boldsymbol{P}(\boldsymbol{x})$;
tries to	$\rightsquigarrow$	$\boldsymbol{try}$;	rapidly	$\rightsquigarrow$	$\lambda \boldsymbol{P}\lambda \boldsymbol{x}.\boldsymbol{rapidly}(\boldsymbol{P},\boldsymbol{x}) \wedge \boldsymbol{P}(\boldsymbol{x})$;
wishes to	$\rightsquigarrow$	$\boldsymbol{wish}$;	voluntarily	$\rightsquigarrow$	$\lambda \boldsymbol{P}\lambda \boldsymbol{x}.\boldsymbol{voluntary}(\boldsymbol{P},\boldsymbol{x}) \wedge \boldsymbol{P}(\boldsymbol{x})$;
possibly	$\rightsquigarrow$	$\lambda \boldsymbol{p}.\Diamond \boldsymbol{p}$;	necessarily	$\rightsquigarrow$	$\lambda \boldsymbol{p}.\Box \boldsymbol{p}$;
t_n/it$_n$	$\rightsquigarrow$	$\boldsymbol{x}_n$, for each n;	a	$\rightsquigarrow$	$\lambda \boldsymbol{P}_1\lambda \boldsymbol{P}\bigvee \boldsymbol{x}.\boldsymbol{P}_1(\boldsymbol{x}) \wedge \boldsymbol{P}(\boldsymbol{x})$;
(s)he$_n$	$\rightsquigarrow$	$\boldsymbol{x}_n$, for each n;	every	$\rightsquigarrow$	$\lambda \boldsymbol{P}_1\lambda \boldsymbol{P}\bigwedge \boldsymbol{x}.\boldsymbol{P}_1(\boldsymbol{x}) \dot{\rightarrow} \boldsymbol{P}(\boldsymbol{x})$;
the	$\rightsquigarrow$	$\lambda \boldsymbol{P}_1\lambda \boldsymbol{P}\bigvee \boldsymbol{x}\bigwedge \boldsymbol{y}.(\boldsymbol{P}_1(\boldsymbol{y}) \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}) \wedge \boldsymbol{P}(\boldsymbol{x})$			

In the third-to-last line from Definition 3.2.2, t_n represents the trace of a moved constituent in a logical form that is translated as a free variable, $\boldsymbol{x}_n$.

In line with Partee's conjecture (cf. Ch. 1.2), Definition 3.2.2 assigns same-type translations to common nouns, intransitive verbs, complementizers, and sentence adverbs (all type $(o;o)$). Determiners and quantifiers are translated through the use of the non-logical TY$_0$ constants $\bigwedge$, $\bigvee$, $\wedge$, $\dot{\rightarrow}$, $\doteq$, and $\dot{\leftrightarrow}$. In all other respects, our single-type translations are very similar to the translations from (**Montague, 1973**), cf. (**Gallin, 1975**).

To support Definition 3.2.2, we show that the logic TY$_0$ translates the example sentences from (**Montague, 1973**, Sect. 4). We begin by translating the logi-

cal form of the sentence Bill walks:

The base rule of type-driven translation assigns the proper name Bill and the intransitive verb walks the type-o constant $\boldsymbol{bill}$ and the type-$(o;o)$ constant $\boldsymbol{walk}$, respectively. Thus, the name Bill is interpreted as a primitive entity. The verb walks is interpreted as a function that applies to entities (e.g. to the entity denoted by the TY_0 translation of Bill) to yield entities (i.e. the entity denoted by the TY_0 translation of Bill walks).

The sentence Bill walks is translated in (3.2.1). In the translation, steps (3) and (4) use the rules of copying (T1) and application (T2) from Definition 3.2.1:

(3.2.1)
1. $[_{\mathrm{NP}}\mathsf{Bill}] \rightsquigarrow \boldsymbol{bill}$
2. $[_{\mathrm{IV}}\mathsf{walks}] \rightsquigarrow \boldsymbol{walk}$
3. $[_{\mathrm{VP}}[_{\mathrm{IV}}\mathsf{walks}]] \rightsquigarrow \boldsymbol{walk}$
4. $[_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{Bill}][_{\mathrm{VP}}[_{\mathrm{IV}}\mathsf{walks}]]] \rightsquigarrow \boldsymbol{walk}(\boldsymbol{bill})$

The translations of the logical forms of the sentences A man walks, Every man walks, The man walks, A price rises, and The temperature rises, cf. (**Montague, 1973**, pp. 266, 268), are given in (3.2.2) to (3.2.6):

(3.2.2) $[_{\mathrm{S}}[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{man}]][[_{\mathrm{IV}}\mathsf{walks}]]] \rightsquigarrow \bigvee \boldsymbol{x}.\boldsymbol{man}(\boldsymbol{x}) \wedge \boldsymbol{walk}(\boldsymbol{x})$

(3.2.3) $[_{\mathrm{S}}[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{every}][\mathsf{man}]][[\mathsf{walks}]]] \rightsquigarrow \bigwedge \boldsymbol{x}.\boldsymbol{man}(\boldsymbol{x}) \dot{\rightarrow} \boldsymbol{walk}(\boldsymbol{x})$

(3.2.4) $[_{\mathrm{S}}[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{the}][_{\mathrm{N}}\mathsf{man}]][[_{\mathrm{IV}}\mathsf{walks}]]] \rightsquigarrow \bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}.(\boldsymbol{man}(\boldsymbol{y}) \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}) \wedge \boldsymbol{walk}(\boldsymbol{x})$

(3.2.5) $[_{\mathrm{S}}[[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{price}]][_{\mathrm{IV}}\mathsf{rises}]] \rightsquigarrow \bigvee \boldsymbol{P}.\boldsymbol{price}(\boldsymbol{P}) \wedge \boldsymbol{rise}(\boldsymbol{P})$

(3.2.6) $[_{\mathrm{S}}[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{the}][_{\mathrm{N}}\mathsf{temperature}]][_{\mathrm{IV}}\mathsf{rises}]] \rightsquigarrow \bigvee \boldsymbol{P} \bigwedge \boldsymbol{P}_1.(\boldsymbol{temp}(\boldsymbol{P}_1) \dot{\leftrightarrow} \boldsymbol{P} \doteq \boldsymbol{P}_1) \wedge \boldsymbol{rise}(\boldsymbol{P})$

The translations of the logical forms from (3.2.5) and (3.2.6) assume a meaning-shifting rule for the determiners a and the. This rule is definable by a TY_0 term, as we show in Appendix D.1. This appendix also contains detailed derivations of the translations from (3.2.5) and (3.2.6), and of the translations of the sentences The temperature is ninety (3.2.21) and Ninety rises (D.1.1). These translations are instrumental in our solution to Partee's temperature puzzle (cf. Sect. 3.3).

In our translations of transitive verbs from Definition 3.2.2, the relevant type-$(o\,o;o)$ constant (e.g. the constant $\boldsymbol{find}$ $(=_\eta \lambda\boldsymbol{y}\lambda\boldsymbol{x}.\boldsymbol{find}(\boldsymbol{y},\boldsymbol{x}))$) reflects the structure of transitive verbs in English. By (PS 6), these verbs combine with their direct object (here, translated '$\boldsymbol{y}$') before combining with their subject ('$\boldsymbol{x}$'). As a result, the TY_0 translation of the verb finds is interpreted as encoding the information that $\boldsymbol{x}$ finds $\boldsymbol{y}$. This fact is reflected in the translation of the logical form of the sentence John finds a unicorn, cf. (**Montague, 1973**, p. 266):

(3.2.7) $[_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{John}][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{finds}][_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{unicorn}]]]]$

$\rightsquigarrow \bigvee \boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{find}(\boldsymbol{x}, \boldsymbol{john})$

To enable the TY$_0$ translation of the result of merging a transitive verb (type $(\underline{(o;o);o)}\ o;o)$) with a proper name (type o), we introduce a variant of Partee's type-shifting rule 'lift' from (**Partee, 1987**, p. 362). This variant is defined below:

DEFINITION 3.2.3. The function $\mathit{lift} := \lambda\boldsymbol{x}\lambda\boldsymbol{P}.\boldsymbol{P}(\boldsymbol{x})$ sends entity-denoting TY$_0$ terms to the designators of functions from (functions from entities to entities) to entities (type $((o;o);o)$).

Our translation of proper names into basic-type TY$_0$ terms (and, hence, the need for the function *lift*) is motivated by the interpretation of sentences and complement phrases in the basic type, and by our wish (in correspondence with Partee's conjecture) to interpret names in the semantic type of sentences.

The function *lift* enables the translation of the logical form of the sentence John finds Mary as follows:

(3.2.8)
1. $[_{\mathrm{NP}}\mathsf{Mary}] \rightsquigarrow \boldsymbol{mary}$
2. $[_{\mathrm{TV}}\mathsf{finds}] \rightsquigarrow \lambda\boldsymbol{Q}\lambda\boldsymbol{x}.\boldsymbol{Q}(\lambda\boldsymbol{y}.\boldsymbol{find}(\boldsymbol{y},\boldsymbol{x}))$
3. $[_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{finds}][_{\mathrm{NP}}\mathsf{Mary}]] \rightsquigarrow \lambda\boldsymbol{Q}\lambda\boldsymbol{x}.\boldsymbol{Q}(\lambda\boldsymbol{y}.\boldsymbol{find}(\boldsymbol{y},\boldsymbol{x}))\,[\mathit{lift}(\boldsymbol{mary})]$
   $= \lambda\boldsymbol{Q}\lambda\boldsymbol{x}.\boldsymbol{Q}(\lambda\boldsymbol{y}.\boldsymbol{find}(\boldsymbol{y},\boldsymbol{x}))\,[\lambda\boldsymbol{x}\lambda\boldsymbol{P}.\boldsymbol{P}(\boldsymbol{x})(\boldsymbol{mary})]$
   $= \lambda\boldsymbol{Q}\lambda\boldsymbol{x}.\boldsymbol{Q}(\lambda\boldsymbol{y}.\boldsymbol{find}(\boldsymbol{y},\boldsymbol{x}))\,[\lambda\boldsymbol{P}.\boldsymbol{P}(\boldsymbol{mary})]$
   $= \lambda\boldsymbol{x}.[\lambda\boldsymbol{P}.\boldsymbol{P}(\boldsymbol{mary})](\lambda\boldsymbol{y}.\boldsymbol{find}(\boldsymbol{y},\boldsymbol{x}))$
   $= \lambda\boldsymbol{x}.[(\lambda\boldsymbol{y}.\boldsymbol{find}(\boldsymbol{y},\boldsymbol{x}))(\boldsymbol{mary})]$
   $= \lambda\boldsymbol{x}.\boldsymbol{find}(\boldsymbol{mary},\boldsymbol{x})$
4. $[_{\mathrm{NP}}\mathsf{John}] \rightsquigarrow \boldsymbol{john}$
5. $[_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{John}][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{finds}][_{\mathrm{NP}}\mathsf{Mary}]]] \rightsquigarrow \boldsymbol{find}(\boldsymbol{mary},\boldsymbol{john})$

The function *lift* also enables the translations of the logical forms[3] of the sentences (1a) and (2a) from Chapter 1.2.1:

(3.2.9) $[_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{Pat}][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{remembers}][_{\mathrm{NP}}\mathsf{Bill}]]] \rightsquigarrow \boldsymbol{remember}(\boldsymbol{bill},\boldsymbol{pat})$

(3.2.10) $[_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{Sherlock}][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{fears}][_{\mathrm{NP}}\mathsf{Moriarty}]]] \rightsquigarrow \boldsymbol{fear}(\boldsymbol{moriarty},\boldsymbol{sherlock})$

However, our TY$_0$-based semantics is not restricted to the translation of one particular class of complements of transitive verbs (by (PS 6), to [[TV][NP]]-forms). It also enables the translation of the results of combining some transitive verbs with complement phrases (cf. (1b), (2b)). This observation is captured below:

[3]Since it is not currently relevant, we neglect the tense and aspect of the original examples.

PROPOSITION 3.1 (NP/CP neutrality). *A 'pure' single-type semantics enables the interpretation of* (*both guises of*) NP/CP *complement-neutral expressions.*

The possibility of interpreting [[TV][CP]]-structures in a 'pure' single-type semantics is witnessed by the TY_0 translations (in (3.2.11), (3.2.12)) of the logical forms of the sentences (1b) and (2b) from Chapter 1.2.1. To accommodate structures of this form, we introduce the additional phrase structure rule (PS 12):

(PS 12) VP $\longrightarrow$ $\mathrm{TV}^{\dagger}$ CP Verb Phrase,
where $\mathrm{TV}^{\dagger} \ni$ remember, fear, be, etc.

The sentences (1b) and (2b) then have the following translation:

(3.2.11)
$[_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{Pat}]^1\ [_{\mathrm{S}}\ t_1\ [_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{remembers}][_{\mathrm{CP}}[_{\mathrm{C}}\mathsf{that}][_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{Bill}][_{\mathrm{VP}}[\mathsf{waits}][_{\mathrm{PP}}[_{\mathrm{P}}\mathsf{for}][_{\mathrm{NP}}\mathsf{she}_1]]]]]]]]$
$\rightsquigarrow \boldsymbol{remember}\big(\boldsymbol{for}(\boldsymbol{pat}, \boldsymbol{wait}, \boldsymbol{bill}), \boldsymbol{pat}\big)$

(3.2.12)
$[_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{Sherlock}]^2\ [_{\mathrm{S}}\ t_2\ [_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{fears}][_{\mathrm{CP}}[_{\mathrm{C}}\mathsf{that}][_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{Moriarty}][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{destroys}][_{\mathrm{NP}}\mathsf{he}_2]]]]]]]$
$\rightsquigarrow \boldsymbol{fear}\big(\boldsymbol{destroy}(\boldsymbol{sherlock}, \boldsymbol{moriarty}), \boldsymbol{sherlock}\big)$

We will see below that our TY_0-based semantics also accommodates NP/CP coordinations and CP equatives (cf. Ch. 1.2.1; (4)–(6), (7a), (7b)). However, before we show this, we first translate the logical forms of some other PTQ sentences.

The translation of the narrow-scope reading of the sentence John seeks a unicorn (in (3.2.13)), cf. (**Montague, 1973**, p. 266), is obtained analogously to the translation from (3.2.8) (without the use of the function *lift*).

(3.2.13) $[_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{John}][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{seeks}][_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{unicorn}]]]]$
$\rightsquigarrow \boldsymbol{seek}\big([\lambda \boldsymbol{P} \bigvee \boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{P}(\boldsymbol{x})], \boldsymbol{john}\big)$

The rule of Quantifier Raising enables the translation of the sentence's wide-scope reading as follows:

(3.2.14) 1. $[_{\mathrm{TV}}\mathsf{seeks}] \rightsquigarrow \boldsymbol{seek} = \lambda \boldsymbol{Q}\, \lambda \boldsymbol{x}.\boldsymbol{seek}(\boldsymbol{Q}, \boldsymbol{x})$

2. $[_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{seeks}]\ t_0] \rightsquigarrow \lambda \boldsymbol{Q}\, \lambda \boldsymbol{x}.\boldsymbol{seek}(\boldsymbol{Q}, \boldsymbol{x})\, [\lambda \boldsymbol{P}.\boldsymbol{P}(\boldsymbol{x}_0)]$
$= \lambda \boldsymbol{x}.\boldsymbol{seek}\big([\lambda \boldsymbol{P}.\boldsymbol{P}(\boldsymbol{x}_0)], \boldsymbol{x}\big)$

3. $[_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{John}][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{seeks}]\ t_0]] \rightsquigarrow \boldsymbol{seek}\big([\lambda \boldsymbol{P}.\boldsymbol{P}(\boldsymbol{x}_0)], \boldsymbol{john}\big)$

4. $[_{\mathrm{S}}[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{unicorn}]]^0\ [_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{John}][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{seeks}]\ t_0]]]$
$\rightsquigarrow \lambda \boldsymbol{P}_1 \bigvee \boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{P}_1(\boldsymbol{x})\big[\lambda \boldsymbol{x}_0.\boldsymbol{seek}\big([\lambda \boldsymbol{P}.\boldsymbol{P}(\boldsymbol{x}_0)], \boldsymbol{john}\big)\big]$
$= \bigvee \boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{seek}\big([\lambda \boldsymbol{P}.\boldsymbol{P}(\boldsymbol{x})], \boldsymbol{john}\big)$

Translations of the object narrow- and wide-scope readings of the sentences John

talks about a unicorn and A woman loves every man, cf. (**Montague, 1973**, pp. 267, 268), are given below:

(3.2.15) $[_{S}[_{NP}\mathsf{John}][_{VP}[_{IV}\mathsf{talks}][_{PP}[_{P}\mathsf{about}][_{NP}[_{DET}\mathsf{a}][_{N}\mathsf{unicorn}]]]]]$

$\rightsquigarrow \quad \boldsymbol{about}\left([\lambda \boldsymbol{P} \bigvee \boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{P}(\boldsymbol{x})], \boldsymbol{talk}, \boldsymbol{john}\right)$

(3.2.16) $[_{S}[_{NP}[_{DET}\mathsf{a}][_{N}\mathsf{unicorn}]]^{0}\,[_{S}[_{NP}\mathsf{John}][_{VP}[_{IV}\mathsf{talks}][_{PP}[_{P}\mathsf{about}]\ t_0]]]]$

$\rightsquigarrow \quad \bigvee \boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{about}\left([\lambda \boldsymbol{P}.\boldsymbol{P}(\boldsymbol{x})], \boldsymbol{talk}, \boldsymbol{john}\right)$

(3.2.17) $[_{S}[_{NP}[_{DET}\mathsf{a}][_{N}\mathsf{woman}]][_{VP}[_{TV}\mathsf{loves}][_{NP}[_{DET}\mathsf{every}][_{N}\mathsf{man}]]]]$

$\rightsquigarrow \quad \bigvee \boldsymbol{x}.\boldsymbol{woman}(\boldsymbol{x}) \wedge \left(\bigwedge \boldsymbol{y}.\boldsymbol{man}(\boldsymbol{y}) \dot{\rightarrow} \boldsymbol{love}(\boldsymbol{y}, \boldsymbol{x})\right)$

(3.2.18) $[[_{NP}[_{DET}\mathsf{every}][_{N}\mathsf{man}]]^{1}[_{S}[_{NP}[_{DET}\mathsf{a}][_{N}\mathsf{woman}]][_{VP}[_{TV}\mathsf{loves}]\ t_1]]]$

$\rightsquigarrow \quad \bigwedge \boldsymbol{y}.\boldsymbol{man}(\boldsymbol{y}) \dot{\rightarrow} \left(\bigvee \boldsymbol{x}.\boldsymbol{woman}(\boldsymbol{x}) \wedge \boldsymbol{love}(\boldsymbol{y}, \boldsymbol{x})\right)$

The logical forms of the sentences Bill is Mary, Bill is a man, and The temperature is ninety, cf. (**Montague, 1973**, pp. 267, 268), have their expected translations:

(3.2.19) $[_{S}[_{NP}\mathsf{Bill}][_{VP}[_{TV}\mathsf{is}][_{NP}\mathsf{Mary}]]] \quad \rightsquigarrow \quad \boldsymbol{bill} \doteq \boldsymbol{mary}$

(3.2.20) $[_{S}[_{NP}\mathsf{Bill}][_{VP}[_{TV}\mathsf{is}][_{NP}[_{DET}\mathsf{a}][_{N}\mathsf{man}]]]]$

$\rightsquigarrow \quad \left(\bigvee \boldsymbol{x}.\boldsymbol{man}(\boldsymbol{x}) \wedge \boldsymbol{bill} \doteq \boldsymbol{x}\right) \quad = \quad \boldsymbol{man}(\boldsymbol{bill})$

(3.2.21) $[_{S}[_{NP}[_{DET}\mathsf{the}][_{N}\mathsf{temperature}]][_{VP}[_{TV}\mathsf{is}][_{NP}\mathsf{ninety}]]]$

$\rightsquigarrow \quad \bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}.\left(\left(\bigvee \boldsymbol{P}.\boldsymbol{temp}(\boldsymbol{P}) \wedge \boldsymbol{y} \doteq \boldsymbol{P}(\boldsymbol{w})\right) \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}\right) \wedge \boldsymbol{x} \doteq \boldsymbol{ninety}$

Notably, because of its same-type interpretation of proper names and complement phrases, our single-type semantics also enables the interpretation of CP equatives. The translation of the logical form of (7a) is given below. This translation involves the application of the function *lift* to the TY$_0$ translation of a *complement* phrase:

(3.2.22)
1. $[_{CP}[_{C}\mathsf{that}][_{S}[_{NP}\mathsf{Mary}][_{VP}[_{TV}\mathsf{hates}][_{NP}\mathsf{Bill}]]]] \rightsquigarrow \boldsymbol{hate}(\boldsymbol{bill}, \boldsymbol{mary})$
2. $[_{TV}\mathsf{is}] \rightsquigarrow \lambda \boldsymbol{Q} \lambda \boldsymbol{x}.\boldsymbol{Q}(\lambda \boldsymbol{y}.\boldsymbol{x} \doteq \boldsymbol{y})$
3. $[_{VP}[_{TV}\mathsf{is}][_{CP}[_{C}\mathsf{that}][_{S}[_{NP}\mathsf{Mary}][_{VP}[_{TV}\mathsf{hates}][_{NP}\mathsf{Bill}]]]]]$

$\rightsquigarrow \lambda \boldsymbol{Q} \lambda \boldsymbol{x}.\boldsymbol{Q}(\lambda \boldsymbol{y}.\boldsymbol{x} \doteq \boldsymbol{y})\,[\mathit{lift}(\boldsymbol{hate}(\boldsymbol{bill}, \boldsymbol{mary}))]$

$= \lambda \boldsymbol{Q} \lambda \boldsymbol{x}.\boldsymbol{Q}(\lambda \boldsymbol{y}.\boldsymbol{x} \doteq \boldsymbol{y})\,[\lambda \boldsymbol{P}.\boldsymbol{P}(\boldsymbol{hate}(\boldsymbol{bill}, \boldsymbol{mary}))]$

$= \lambda \boldsymbol{x}.[\lambda \boldsymbol{P}.\boldsymbol{P}(\boldsymbol{hate}(\boldsymbol{bill}, \boldsymbol{mary}))](\lambda \boldsymbol{y}.\boldsymbol{x} \doteq \boldsymbol{y})$

$= \lambda \boldsymbol{x}.[(\lambda \boldsymbol{y}.\boldsymbol{x} \doteq \boldsymbol{y})(\boldsymbol{hate}(\boldsymbol{bill}, \boldsymbol{mary}))]$

$= \lambda \boldsymbol{x}.\boldsymbol{x} \doteq \boldsymbol{hate}(\boldsymbol{bill}, \boldsymbol{mary})$

4. $[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{the}][_{\mathrm{N}}\mathsf{problem}]] \rightsquigarrow \lambda \boldsymbol{P} \bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}.(\boldsymbol{problem}(\boldsymbol{y}) \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}) \wedge \boldsymbol{P}(\boldsymbol{x})$
5. $[_{\mathrm{S}}[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{the}][_{\mathrm{N}}\mathsf{problem}]][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{is}][_{\mathrm{CP}}[_{\mathrm{C}}\mathsf{that}][_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{Mary}][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{hates}][_{\mathrm{NP}}\mathsf{Bill}]]]]]]$

$$\rightsquigarrow \lambda \boldsymbol{P} \bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}.(\boldsymbol{problem}(\boldsymbol{y}) \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}) \wedge \boldsymbol{P}(\boldsymbol{x})\,[\lambda \boldsymbol{z}.\boldsymbol{z} \doteq \boldsymbol{hate}(\boldsymbol{bill}, \boldsymbol{mary})]$$

$$= \bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}.(\boldsymbol{problem}(\boldsymbol{y}) \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}) \wedge [\lambda \boldsymbol{z}.\boldsymbol{z} \doteq \boldsymbol{hate}(\boldsymbol{bill}, \boldsymbol{mary})]\,(\boldsymbol{x})$$

$$= \bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}.(\boldsymbol{problem}(\boldsymbol{y}) \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}) \wedge \boldsymbol{x} \doteq \boldsymbol{hate}(\boldsymbol{bill}, \boldsymbol{mary})$$

This completes our translation of PTQ words and atomic sentences into terms of the logic TY_0. The next section discusses the TY_0 translation of molecular sentences from Montague's PTQ fragment (e.g. of the sentence John finds and eats a unicorn, cf. (**Montague, 1973**, p. 269)). We will see that the same-type interpretation of proper names and complement phrases in our o-based semantics enables an accommodation of the coordination phenomena from Chapter 1.2.1.

3.2.4. Translations of Molecular Logical Forms. To enable the compositional translation of coordinated logical forms, we only need to specify the TY_0 translations of the linguistic connectives from the PTQ fragment. For completeness, we also supply the TY_0 translation of the adverb of negation not (or of the expression it is not the case that). These translations are provided in Definition 3.2.4. In the definition, $\boldsymbol{R}$ and $\boldsymbol{R}_1$ are variables of the type $(\alpha_1 \ldots \alpha_n; o)$, where $\alpha_1, \ldots, \alpha_n \in \mathsf{0Type}$ and $0 \leq n \in \mathbb{N}$. We assume that $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ (abbrev. $\vec{\boldsymbol{X}}$) is a sequence of variables of the types $\alpha_1, \ldots, \alpha_n$.

Definition 3.2.4 (TY_0 translations of PTQ connectives). The rule (T0) associates the linguistic connectives and, or, and not with the following TY_0 terms:

(i)	and	$\rightsquigarrow$	$\lambda \boldsymbol{R}_1 \lambda \boldsymbol{R} \lambda \vec{\boldsymbol{X}}.\boldsymbol{R}(\vec{\boldsymbol{X}}) \wedge \boldsymbol{R}_1(\vec{\boldsymbol{X}})$	(Conjunction);
(ii)	or	$\rightsquigarrow$	$\lambda \boldsymbol{R}_1 \lambda \boldsymbol{R} \lambda \vec{\boldsymbol{X}}.\boldsymbol{R}(\vec{\boldsymbol{X}}) \vee \boldsymbol{R}_1(\vec{\boldsymbol{X}})$	(Disjunction);
(iii)	not	$\rightsquigarrow$	$\lambda \boldsymbol{R} \lambda \vec{\boldsymbol{X}}.\neg \boldsymbol{R}(\vec{\boldsymbol{X}})$	(Negation)

The TY_0 translations from Definition 3.2.4 are flexible, such that they are defined for the TY_0 translations of expressions from different syntactic categories. Thus, the conjunction and connects sentences and complement phrases (e.g. Bill walks and Mary talks, that Bill walks and that Mary talks), proper names (Bill and Mary), (in-)transitive verbs (walks and talks, finds and eats), and many other kinds of expressions. The coordination of proper names and quantified NPs (e.g. Bill and every woman) is enabled by the type-shifting operation *lift* (cf. Def. 3.2.3).

Items (3.2.23) and (3.2.24) (next page) illustrate the use of the conjunction and in the coordination of the sentences Bill walks and Every man finds a unicorn, and of the verbs finds and eats:

(3.2.23) 1. $[_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{Bill}][_{\mathrm{VP}}[_{\mathrm{IV}}\mathsf{walks}]]] \rightsquigarrow \boldsymbol{walk}(\boldsymbol{bill})$

2. $[_{\mathrm{S}}[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{every}][_{\mathrm{N}}\mathsf{man}]][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{finds}][_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{unicorn}]]]]$
$\rightsquigarrow \bigwedge \boldsymbol{y}.\boldsymbol{man}(\boldsymbol{y}) \dot{\rightarrow} (\bigvee \boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{find}(\boldsymbol{y},\boldsymbol{x}))$

3. $[_{\mathrm{CONJ}}\mathsf{and}] \rightsquigarrow \lambda\boldsymbol{q}\lambda\boldsymbol{p}.\boldsymbol{p} \wedge \boldsymbol{q}$

4. $[[_{\mathrm{CONJ}}\mathsf{and}][_{\mathrm{S}}[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{every}][_{\mathrm{N}}\mathsf{man}]][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{finds}][_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{unicorn}]]]]]$
$\rightsquigarrow \lambda\boldsymbol{q}\lambda\boldsymbol{p}.\boldsymbol{p} \wedge \boldsymbol{q}\,[\bigwedge \boldsymbol{y}.\boldsymbol{man}(\boldsymbol{y}) \dot{\rightarrow} (\bigvee \boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{find}(\boldsymbol{y},\boldsymbol{x}))]$
$\rightsquigarrow \lambda\boldsymbol{p}.\boldsymbol{p} \wedge (\bigwedge \boldsymbol{y}.\boldsymbol{man}(\boldsymbol{y}) \dot{\rightarrow} (\bigvee \boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{find}(\boldsymbol{y},\boldsymbol{x})))$

5. $[_{\mathrm{S}}[_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{Bill}][_{\mathrm{VP}}[_{\mathrm{IV}}\mathsf{walks}]]]$
$[[_{\mathrm{CONJ}}\mathsf{and}][_{\mathrm{S}}[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{every}][_{\mathrm{N}}\mathsf{man}]][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{finds}][_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{unicorn}]]]]]]$
$\rightsquigarrow \boldsymbol{walk}(\boldsymbol{bill}) \wedge$
$(\bigwedge \boldsymbol{y}.\boldsymbol{man}(\boldsymbol{y}) \dot{\rightarrow} (\bigvee \boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{find}(\boldsymbol{y},\boldsymbol{x})))$

(3.2.24) 1. $[_{\mathrm{TV}}\mathsf{finds}] \rightsquigarrow \lambda\boldsymbol{Q}\lambda\boldsymbol{x}.\boldsymbol{Q}(\lambda\boldsymbol{y}.\boldsymbol{find}(\boldsymbol{y},\boldsymbol{x}))$

2. $[_{\mathrm{TV}}\mathsf{eats}] \rightsquigarrow \lambda\boldsymbol{Q}\lambda\boldsymbol{x}.\boldsymbol{Q}(\lambda\boldsymbol{y}.\boldsymbol{eat}(\boldsymbol{y},\boldsymbol{x}))$

3. $[_{\mathrm{CONJ}}\mathsf{and}] \rightsquigarrow \lambda\boldsymbol{L}_1\lambda\boldsymbol{L}\lambda\boldsymbol{Q}\lambda\boldsymbol{x}.\boldsymbol{L}(\boldsymbol{Q},\boldsymbol{x}) \wedge \boldsymbol{L}_1(\boldsymbol{Q},\boldsymbol{x})$

4. $[[_{\mathrm{CONJ}}\mathsf{and}][_{\mathrm{TV}}\mathsf{eats}]]$
$\rightsquigarrow \lambda\boldsymbol{L}_1\lambda\boldsymbol{L}\lambda\boldsymbol{Q}\lambda\boldsymbol{x}.\boldsymbol{L}(\boldsymbol{Q},\boldsymbol{x}) \wedge \boldsymbol{L}_1(\boldsymbol{Q},\boldsymbol{x})\,[\lambda\boldsymbol{Q}_1\lambda\boldsymbol{z}.\boldsymbol{Q}_1(\lambda\boldsymbol{y}.\boldsymbol{eat}(\boldsymbol{y},\boldsymbol{z}))]$
$= \lambda\boldsymbol{L}\lambda\boldsymbol{Q}\lambda\boldsymbol{x}.\boldsymbol{L}(\boldsymbol{Q},\boldsymbol{x}) \wedge [\lambda\boldsymbol{Q}_1\lambda\boldsymbol{z}.\boldsymbol{Q}_1(\lambda\boldsymbol{y}.\boldsymbol{eat}(\boldsymbol{y},\boldsymbol{z}))](\boldsymbol{Q},\boldsymbol{x})$
$= \lambda\boldsymbol{L}\lambda\boldsymbol{Q}\lambda\boldsymbol{x}.\boldsymbol{L}(\boldsymbol{Q},\boldsymbol{x}) \wedge \boldsymbol{Q}(\lambda\boldsymbol{y}.\boldsymbol{eat}(\boldsymbol{y},\boldsymbol{x}))$

5. $[_{\mathrm{TV}}[_{\mathrm{TV}}\mathsf{finds}][[_{\mathrm{CONJ}}\mathsf{and}][_{\mathrm{TV}}\mathsf{eats}]]]$
$\rightsquigarrow \lambda\boldsymbol{L}\lambda\boldsymbol{Q}\lambda\boldsymbol{x}.\boldsymbol{L}(\boldsymbol{Q},\boldsymbol{x}) \wedge \boldsymbol{Q}(\lambda\boldsymbol{y}.\boldsymbol{eat}(\boldsymbol{y},\boldsymbol{x}))$
$[\lambda\boldsymbol{Q}_1\lambda\boldsymbol{z}.\boldsymbol{Q}(\lambda\boldsymbol{y}.\boldsymbol{find}(\boldsymbol{y},\boldsymbol{z}))]$
$= \lambda\boldsymbol{Q}\lambda\boldsymbol{x}.[\lambda\boldsymbol{Q}_1\lambda\boldsymbol{z}.\boldsymbol{Q}(\lambda\boldsymbol{y}.\boldsymbol{find}(\boldsymbol{y},\boldsymbol{z}))](\boldsymbol{Q},\boldsymbol{x}) \wedge \boldsymbol{Q}(\lambda\boldsymbol{y}.\boldsymbol{eat}(\boldsymbol{y},\boldsymbol{x}))$
$= \lambda\boldsymbol{Q}\lambda\boldsymbol{x}.\boldsymbol{Q}(\lambda\boldsymbol{y}.\boldsymbol{find}(\boldsymbol{y},\boldsymbol{x})) \wedge \boldsymbol{Q}(\lambda\boldsymbol{y}.\boldsymbol{eat}(\boldsymbol{y},\boldsymbol{x}))$
$= \lambda\boldsymbol{Q}\lambda\boldsymbol{x}.\boldsymbol{Q}(\lambda\boldsymbol{y}.(\boldsymbol{find} \wedge \boldsymbol{eat})(\boldsymbol{y},\boldsymbol{x}))$

6. $[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{unicorn}]] \rightsquigarrow \lambda\boldsymbol{P}\bigvee \boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{P}(\boldsymbol{x})$

7. $[_{\mathrm{VP}}[_{\mathrm{TV}}[_{\mathrm{TV}}\mathsf{finds}][[_{\mathrm{CONJ}}\mathsf{and}][_{\mathrm{TV}}\mathsf{eats}]]][_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{unicorn}]]]$
$\rightsquigarrow \lambda\boldsymbol{Q}\lambda\boldsymbol{z}.\boldsymbol{Q}(\lambda\boldsymbol{y}.(\boldsymbol{find} \wedge \boldsymbol{eat})(\boldsymbol{y},\boldsymbol{z}))\,[\lambda\boldsymbol{P}\bigvee \boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{P}(\boldsymbol{x})]$
$= \lambda\boldsymbol{z}.[\lambda\boldsymbol{P}\bigvee \boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{P}(\boldsymbol{x})](\lambda\boldsymbol{y}.(\boldsymbol{find} \wedge \boldsymbol{eat})(\boldsymbol{y},\boldsymbol{z}))$

$$= \lambda z \bigvee x.\, \boldsymbol{unicorn}(x) \wedge (\lambda y.(\boldsymbol{find} \wedge \boldsymbol{eat})(y, z))(x)$$

$$= \lambda y \bigvee x.\, \boldsymbol{unicorn}(x) \wedge (\boldsymbol{find} \wedge \boldsymbol{eat})(x, y)$$

8. $[_{S}[_{NP}\mathsf{John}][_{VP}[_{TV}[_{TV}\mathsf{finds}][[_{CONJ}\mathsf{and}][_{TV}\mathsf{eats}]]][_{NP}[_{DET}\mathsf{a}][_{N}\mathsf{unicorn}]]]]$

$$\rightsquigarrow \bigvee x.\, \boldsymbol{unicorn}(x) \wedge (\boldsymbol{find} \wedge \boldsymbol{eat})(x, \boldsymbol{john})$$

The neutrality of the type o between Montague's types e and $\langle s, t\rangle$ enables the translation of coordinations with a proper name- and a CP-conjunct. In particular, Definition 3.2.4 allows the translation of the results (in (4), (5)) of coordinating the name and the CP in the complements of the occurrences of the verb remembers from (3.2.9) and (3.2.11) (cf. (1a), (1b)), and of the verb fears from (3.2.10) and (3.2.12) (cf. (2a), (2b)):

(3.2.25) $[_{S}[_{NP}\mathsf{Pat}]^1\ [_{S}\ t_1\ [_{VP}[_{TV}\mathsf{remembers}][_{NP}\mathsf{Bill}]$
$[[_{CONJ}\mathsf{and}][_{CP}[_{C}\mathsf{that}][_{S}[_{NP}\mathsf{Bill}][_{VP}[_{IV}\mathsf{waits}][_{PP}[_{P}\mathsf{for}][_{NP}\mathsf{she}_1]]]]]]]]]$

$$\rightsquigarrow \boldsymbol{remember}\big(\big(\boldsymbol{bill} \wedge \boldsymbol{for}(\boldsymbol{pat}, \boldsymbol{wait}, \boldsymbol{bill})\big), \boldsymbol{pat}\big)$$

(3.2.26) $[_{S}[_{NP}\mathsf{Sherlock}]^2\ [_{S}\ t_2\ [_{VP}[_{TV}\mathsf{fears}][_{NP}\mathsf{Moriarty}]$
$[[_{CO}\mathsf{and}][_{CP}[_{C}\mathsf{that}][_{S}[_{NP}\mathsf{Moriarty}][_{VP}[_{TV}\mathsf{destroys}][_{NP}\mathsf{he}_2]]]]]]]]$

$$\rightsquigarrow \boldsymbol{fear}\big(\big(\boldsymbol{moriarty} \wedge \boldsymbol{destroy}(\boldsymbol{sherlock}, \boldsymbol{moriarty})\big), \boldsymbol{sherlock}\big)$$

The possibility of extending coordination to proper names in our TY_0-based semantics is captured in Proposition 3.2:

PROPOSITION 3.2 (NP/CP coordinability). *A 'pure' single-type semantics can model logical forms which contain coordinations with a proper name- and a* CP*-conjunct.*

The use of the connective $\vdash$ in the translation of the logical form of the sentence John does not find a unicorn is demonstrated below:

(3.2.27) 1. $[_{VP}[_{TV}\mathsf{find}][_{NP}[_{DET}\mathsf{a}][_{N}\mathsf{unicorn}]]] \rightsquigarrow \lambda y \bigvee x.\, \boldsymbol{unicorn}(x) \wedge \boldsymbol{find}(x, y)$

2. $[_{ADV}\mathsf{does\ not}] \rightsquigarrow \lambda \boldsymbol{P}\, \lambda x. \vdash \boldsymbol{P}(x)$

3. $[_{VP}[_{ADV}\mathsf{does\ not}][_{VP}[_{TV}\mathsf{find}][_{NP}[_{DET}\mathsf{a}][_{N}\mathsf{unicorn}]]]]$

$$\rightsquigarrow \lambda \boldsymbol{P}\, \lambda y. \vdash \boldsymbol{P}(y)\ [\lambda z \bigvee x.\, \boldsymbol{unicorn}(x) \wedge \boldsymbol{find}(x, z)]$$

$$= \lambda y. \vdash \big(\bigvee x.\, \boldsymbol{unicorn}(x) \wedge \boldsymbol{find}(x, y)\big)$$

4. $[_{S}[_{NP}\mathsf{John}][_{VP}[_{ADV}\mathsf{does\ not}][_{VP}[_{TV}\mathsf{find}][_{NP}[_{DET}\mathsf{a}][_{N}\mathsf{unicorn}]]]]]$

$$\rightsquigarrow \vdash \big(\bigvee x.\, \boldsymbol{unicorn}(x) \wedge \boldsymbol{find}(x, \boldsymbol{john})\big)$$

This completes our translation of logical PTQ forms. To evaluate the ability of our 'pure' single-type semantics to model the phenomena from Proposition 1.3, we next define the notion of equivalence on logical forms of the fragment.

3.3. PTQ Equivalence and Consequence

The possibility of identifying semantically equivalent PTQ expressions in our TY$_0$-based semantics is warranted by the interpretation of these expressions into elements of TY$_0$ models. In particular, since all logical PTQ forms allow a systematic translation into TY$_0$ terms, we can define the equivalence of logical PTQ forms in terms of the mutual entailment of their TY$_0$ translations.

In the definition of equivalence, we let $\boldsymbol{A}_o$ and $\boldsymbol{B}_o$ be the TY$_0$ translations of the logical forms X, respectively Y, such that $X \rightsquigarrow \boldsymbol{A}$ and $Y \rightsquigarrow \boldsymbol{B}$. The semantic equivalence of X and Y is then defined as follows:

DEFINITION 3.3.1 (TY$_0$-based linguistic equivalence). The form X is *equivalent* to Y in the TY$_0$ metatheory w.r.t. the TY$_0$ model $M_{\mathcal{F}}$ and assignment $g_{\mathcal{F}}$, i.e. MEANS$_{M_{\mathcal{F}}}(Y, X)$, if $\models_g \boldsymbol{A} = \boldsymbol{B}$ in all relevant[4] metatheoretical models and assignments.

The above definition supports our intuitions about linguistic equivalence. For example, the equivalence of the TY$_0$ terms $\boldsymbol{remember}(\boldsymbol{bill}, \boldsymbol{pat})$ and $\neg\neg\boldsymbol{remember}(\boldsymbol{bill}, \boldsymbol{pat})$ in the metatheory of the logic TY$_0$ supports the equivalence of the logical forms of the sentences Pat remembers Bill and It is not the case that Pat does not remember Bill. An analogous observation can be made with respect to the equivalence of the logical form of the negation of the sentence Pat remembers Bill and that Bill waits for her and of the logical form of the sentence Pat does not remember Bill or does not remember that Bill waits for her.

Since our TY$_0$-based semantics interprets proper names in the basic type o, Definition 3.3.1 extends the definition of linguistic equivalence to names. As a result, we expect that it be possible, for every pair of names – or for every pair of a name and a sentence (or CP) –, to determine whether its members are equivalent.

However, in practice, the relation of TY$_0$-based equivalence is restricted to pairs of logical forms of same-category expressions whose members receive an interpretation as 'algebraically related' objects (in the sense of Ch. 2.2.2; cf. Ch. 2.4). Examples of such pairs are the pairs of logical forms from the second-to-last paragraph, and pairs consisting of forms like Bill and Bill and (Bill or John). The restriction of TY$_0$-based equivalence to pairs of this form is due to the semantic primitiveness of entities. As a result, unless we have full knowledge of the ordering

[4]The *relevance* of meta- for object-theoretic models concerns the *identification* of models of the object theory with *restricted* models of the metatheory, whose frames and interpretation functions are limited to TY$_0$ terms and objects. This relation will be further discussed in Ch. 7.2.1.

on $\mathcal{O}$ in the designated model, we do not know whether the informativeness relation obtains between 'algebraically unrelated' entities in this model (e.g. between the name Barbara Partee and the sentence Barbara Partee enters the room, or Barbara Partee arrives[5]). But our accommodation of Proposition 1.3.ii requires exactly the equivalence of 'algebraically unrelated' logical forms from these different categories.

The consequence of the above observation is summarized in Proposition 3.3:

PROPOSITION 3.3 (Absence of sentential name-equivalents). *A 'pure' single-type semantics is unable to identify equivalence relations between proper names and sentences.*

Because of the difficulty of obtaining truth- (or falsity-)conditions for basic-type TY_0 terms (cf. Ch. 2.5), our TY_0-based semantics further does not easily accommodate Proposition 1.3.i:

PROPOSITION 3.4 (TY_0 truth-inaptness of names). *In a 'pure' single-type semantics, names and sentences do not have (obvious) truth- and falsity-conditions.*

The inability to model the above phenomena is a significant impediment of the presented single-type semantics, which we will attempt to resolve in the remainder of this dissertation. However, before we do so, we briefly specify the relation of logical consequence on PTQ forms. We will show that, as a result of the definition of TY_0 consequence, our single-type semantics blocks the generation of Partee's temperature puzzle from (**Montague, 1973**, pp. 267–268).

Since the notion of logical consequence is inherently linked to *sentences* – and since we are unaware of any evidence for sentential *entailments* of proper names –, we restrict our definition of linguistic entailment to the syntactic category S. The entailment relation on logical sentence forms is defined below. In the definition, we let $\Xi = \{X \mid X \rightsquigarrow \gamma\}$ and $\Upsilon = \{Y \mid Y \rightsquigarrow \delta\}$ be sets of sentences which are translated into the sets of TY_0 terms $\Gamma = \{\gamma \mid \gamma \in T_o\}$ and $\Delta = \{\delta \mid \delta \in T_o\}$.

DEFINITION 3.3.2 (TY_0-based linguistic entailment). The set of logical forms Ξ *entails* the set of forms Υ in the TY_0 metatheory w.r.t. the TY_0 model $M_{\mathcal{F}}$ and assignment $g_{\mathcal{F}}$, i.e. $\textsc{follow}_{M_{\mathcal{F}}}(\Upsilon, \Xi)$, if $\models_g \Gamma \Rightarrow \Delta$ in all relevant metatheoretical models under all relevant assignments.

Definition 3.3.2 supports our intuitions about the validity of linguistic inferences. Specifically, since the terms $\boldsymbol{remember}(\boldsymbol{bill}, \boldsymbol{pat})$ and $\boldsymbol{remember}\big(\boldsymbol{for}(\boldsymbol{pat}, \boldsymbol{wait}, \boldsymbol{bill}), \boldsymbol{pat}\big)$ are TY_0-deducible from the term $\boldsymbol{remember}\big(\big(\boldsymbol{bill} \wedge \boldsymbol{for}(\boldsymbol{pat}, \boldsymbol{wait}, \boldsymbol{bill})\big), \boldsymbol{pat}\big)$, the inference from the logical form of the sentence Pat remembers Bill and that Bill is waiting for her (cf. (4)) to the logical forms of the sentences Pat remembers Bill (cf. (1a)) and Pat remembers that Bill waits for her (cf. (1b)) is valid.

[5]For simplicity, we again neglect the aspect of the original examples (cf. (9b), (9c)).

A similar observation holds for the validity of the inference from the conjunction of the sentences Bill is the man and The man walks (cf. (3.2.4)) to the sentence Bill walks (cf. (3.2.1)). The TY_0 translation of this inference is given below:[6]

$$\frac{\bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}.(\boldsymbol{man}(\boldsymbol{y}) \leftrightarrows \boldsymbol{x} \doteq \boldsymbol{y}) \wedge\!\!\!\wedge \boldsymbol{x} \doteq \boldsymbol{bill} \quad \bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}.(\boldsymbol{man}(\boldsymbol{y}) \leftrightarrows \boldsymbol{x} \doteq \boldsymbol{y}) \wedge\!\!\!\wedge \boldsymbol{walk}(\boldsymbol{x})}{\boldsymbol{walk}(\boldsymbol{bill})} \tag{3.3.1}$$

Our TY_0 translations of intensional intransitive verbs (e.g. rise, change) and common nouns (temperature, price) prevent the generation of Partee's 'temperature puzzle' in our TY_0-based semantics, cf. (**Montague, 1973**, pp. 267–268). Specifically, since the TY_0 translations of the logical forms of the sentences The temperature is ninety (cf. (3.2.21)) and The temperature rises (cf. (3.2.6)) attribute the properties 'is ninety' and 'rise' to temperature-objects of the types o and $\underline{(o;o)}$, respectively, the TY_0 translation of the sentence Ninety rises cannot be obtained by replacing the type-$(o;o)$ temperature-object $\boldsymbol{P}$ by the entity 'ninety' in the translation of the sentence The temperature rises.

$$\frac{\bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}.\big(\big(\bigvee \boldsymbol{P}.\boldsymbol{temp}(\boldsymbol{P}) \wedge\!\!\!\wedge \boldsymbol{y} \doteq \boldsymbol{P}(\boldsymbol{w})\big) \leftrightarrows \boldsymbol{x} \doteq \boldsymbol{y}\big) \wedge\!\!\!\wedge \boldsymbol{x} \doteq \boldsymbol{ninety} \quad \not{/}\not{/}\not{/}\not{/}\not{/} \bigvee \boldsymbol{P} \bigwedge \boldsymbol{P}_1.(\boldsymbol{temp}(\boldsymbol{P}_1) \leftrightarrows \boldsymbol{P} \doteq \boldsymbol{P}_1) \wedge\!\!\!\wedge \boldsymbol{rise}(\boldsymbol{P}) \not{/}\not{/}\not{/}\not{/}\not{/}}{\bigvee \boldsymbol{P}.\boldsymbol{rise}(\boldsymbol{P}) \wedge\!\!\!\wedge \boldsymbol{P}(\boldsymbol{w}) \doteq \boldsymbol{ninety}} \tag{3.3.2}$$

This completes our discussion of the notions of TY_0-based PTQ equivalence and entailment. We close the chapter with an observation about the explanatory power of the presented single-type semantics.

3.4. Explanatory Power of 'Pure' Single-Type Semantics

The last two sections have demonstrated the ability of our TY_0-based semantics to accommodate NP/CP complement-neutral verbs, CP equatives, and name/CP coordinations. We have attributed this ability to the neutralization of the distinction between Montague's basic types e and $\langle s,t\rangle$, and to the resulting existence of fewer typing constraints on semantic merging. However, the greater tolerance of merging reduces the suitability of TY_0 types as a formal basis for syntactic categories (cf. Sect. 1.1.1).

In Section 3.2.3, we have already suggested that the same-type interpretation of proper names and complement phrases precludes the semantic distinction between verbs which select only NPs and verbs which select only CPs as their com-

[6]For better readability, we will hereafter adopt a sequence-like notation, where the symbol $\Rightarrow$ is replaced by a horizontal line. Terms above (resp. below) this line are the conjuncts of the antecedent (resp. the consequent) of the inference.

plement.[7] Consequently, unless we restrict the phrase structure rule (PS 12) to a proper subset of the category TV, we will no longer be able to explain the ill-formedness[8] of the expression from (16b):

(16) a. Bill eats [$_{NP}$a unicorn].
b. * Bill eats [$_{CP}$that a unicorn exists].

Worse yet, the same-type interpretation of proper names, sentences, and complement phrases blocks a semantic motivation of *most* phrase structure rules from Table 3.2. Thus, in a 'pure' single-type semantics, we will be unable to provide a semantic explanation for the ill-formedness of the expressions from (17b), (18b) or (18c), and (19b) (cf. (PS 7), (PS 1), resp. (PS 2)), or of the expressions from (20b) and (21b) or (21c) (cf. (PS 4), resp. (PS 11)):

(17) a. Bill asserts [$_{NP}$that Mary talks].
b. * Bill asserts [$_{NP}$Mary].

(18) a. [$_{NP}$Bill] exists.
b. * [$_{CP}$That Bill walks] exists.
c. * [$_{S}$Bill walks] exists.

(19) a. Possibly [$_{S}$Bill walks].
b. * Possibly [$_{NP}$Bill].

(20) a. John proves that [$_{S}$Goldbach's Conjecture is correct].
b. * John proves that [$_{NP}$Goldbach's Conjecture].

(21) a. John talks about [$_{NP}$a unicorn].
b. * John talks about [$_{CP}$that he finds a unicorn].
c. * John talks about [$_{S}$he finds a unicorn].

Figure 3.2 (next page) illustrates the 'mismatch' between syntactic categories and 'pure' single-type semantic types.[9] In the figure, the coarser grain of semantic distinctions with respect to their syntactic counterparts is represented by the thickness of the vertical (light and dark grey) bars. These bars cover members of the same category, respectively type.

---

[7] Sag et al. (**2003**) attribute members of these different subclasses the lexical types *v-np-tr* and *v-s-tr*, respectively. Verbs which select both NPs and CPs as their complement are attributed the lexical type *v-nominal-tr*.

[8] Below, ill-formed (or ungrammatical) expressions are marked by an asterisk, '*'.

[9] Notably, expressions of the categories C and SAV also have a same-type interpretation in the Montagovian system. Thus, our semantics only distinguishes itself w.r.t. the interpretations of sentences and proper names, resp. of common nouns and complementizers (or sentence adverbs).

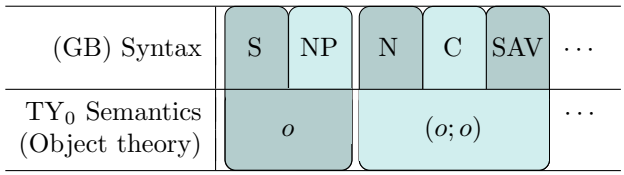

FIGURE 3.2. Syntactic categories and TY_0 types.

The simplest solution to our inability to explain the ill-formedness of (17b) to (21b) (and of (18c) and (21c)) in our 'pure' single-type semantics would be to dismiss a semantic motivation for syntactic categories, and to delegate the explanatory claim of our semantics to the level of syntax. This solution involves the adoption of a traditional theory of syntax (with distinct categories for noun phrases, CPs, and sentences; cf. Sect. 3.1), and the restriction of the arguments of TY_0 translations of lexical elements to translations of logical forms of expressions from the 'right' (or admissible) syntactic category. To block the formation of forms like (17b) to (21b) (cf. (18c), (21c)), we thus restrict the arguments of the TY_0 translations of non-neutral verb phrases and prepositions to TY_0 translations of *noun phrases*, and restrict the arguments of the TY_0 translations of non-neutral sentence adverbs and complementizers to translations of the logical forms of *sentences*.

Yet, the concession of explanatory power to the level of syntax undermines Montague's assumption of the primacy of *semantics* (cf. Ch. 1.1.1). In Chapter 8.3 (cf. Ch. 7.4), we will present an alternative, semantic solution to the above problem which uses a variant of Montague's account of well-formedness. Since the single-type semantics from Chapter 8 shares the full empirical scope of Montague semantics, it combines the explanatory power of the semantics from Section 3.2 (cf. Ch. 1.2.1) with the greater modeling power of the Montagovian system.

3.5. Summary

This chapter has shown that Partee's conjecture from Proposition 1.2 is (almost trivially) supported by a 'pure' single-type semantics which interprets the translations of logical PTQ forms into constructions out of primitive entities: Since this semantics replaces Montague's types for individuals and propositions by the single basic type o, proper names, sentences, and complement phrases all receive an interpretation in this type. The resulting semantics further accommodates NP/CP complement-neutral verbs, NP/CP coordinations, and CP equatives.

However, the chapter has also demonstrated the (methodo-)logical *cost* of the restriction to a single basic type: In particular, the neutralization of the distinction between the semantic types for proper names and sentences reduces the explanatory power of the presented single-type semantics (w.r.t. the explanatory po-

wer of Montague semantics). The absence of the truth-value type t in the TY_0 type system blocks the use of the familiar quantifiers and logical connectives, disables an easy TY_0 truth-definition, and, thus, prevents the truth-evaluation of proper names and sentences. As a result, our 'pure' single-type semantics fails to accommodate Proposition 1.3.i. Because of the primitiveness of the single basic type (s.t. the type o is not identified with any particular Montague type), our semantics further proves unable to identify a name's semantically equivalent sentences (cf. Prop. 1.3.ii). The next part of this dissertation proposes a solution to these problems.

Part II

Single-Type Ontology

The previous chapter has demonstrated the inability of our 'pure' single-type semantics to provide easy truth- and falsity-conditions for logical PTQ forms of the basic type (cf. Prop. 3.4) and to identify semantic equivalence relations between proper names and sentences (cf. Prop. 3.3). The remainder of this dissertation attempts to compensate for these shortcomings. To this aim, we will first identify a suitable Montague type (as introduced in Ch. 1.1.1) which interprets the type o from Part I (step 1). We will then adapt all relevant definitions from Part I to objects of this type (step 2). The present part is concerned with the implementation of step 1. Step 2 will be implemented in Part III.

The rationale behind our particular strategy for the accommodation of Proposition 1.3 is as follows: As we have shown in Chapters 2 and 3, the inability of our TY_0-based semantics to specify an LF's truth- or equivalence-conditions is a consequence of the primitiveness of the single type o (in particular, of the non-identity of o with a construction to the type t; cf. Ch. 2.5). This observation leads us to expect that any Montague type of the form $\langle\alpha_1, \langle \ldots, \langle\alpha_n, t\rangle\rangle\rangle$ which enables a principled[10] representation of individuals and propositions will therewith also model the semantic behavior of proper names and sentences from Proposition 1.3.

The adoption of an 'o-defining' Montague type will further help justify the algebraic structure on the domain of the single basic type, and will obviate the metatheoretic axioms for the behavior of the single-type stand-ins for the familiar truth-functional connectives and quantifiers. In our definition of general single-type models from Section 2.2, we have already observed that some axioms (esp. the axioms for the behavior of negation) rely on the particular semantic analysis of the type o. The identification of suitable single basic types from Chapter 4 provides two such analyses.

Beyond formal reasons, the identification of the type o with a concrete Montague type lends our new semantics intuitive content. From a didactic point of view, it often helps to understand the new in terms of the old (or familiar). Thus, Partee (**2006**) associates her single basic type p with the type for properties of Kratzer-style situations, and describes the single-type correspondents of individuals and propositions as the property of being the minimal situation containing that individual, respectively as the property of being a minimal situation at which the proposition's designator is true (cf. Ch. 1.2.2).

This part of the dissertation prepares two versions of a Partee-style single-type semantics which interpret the single basic type as a particular Montague type. The part is organized as follows: To enable a particular interpretation of the type o from Part I, Chapter 4 identifies a suitable single basic type (or types), and describes the representation of Montagovian individuals and propositions in these types.

[10] Below, *principled* will be defined as satisfying the semantic requirements from Chapter 4.1.

The metaproperties of these types and several associated methodological issues will be discussed in Chapter 5.

To give a conceptual motivation for our single-type choice(s) – and to avoid losing the reader in a blizzard of technical details –, Chapters 4 and 5 will have a largely *informal* character. As a result, many semantic notions (e.g. an individual's *existence* in an index, and a proposition's *aboutness* with respect to a given individual) will be understood in their intuitive pre-theoretical sense. Formal definitions of these notions, which capture the notions' intuitive content, will be provided in Chapter 6.

CHAPTER 4

Histoire d'o

This chapter identifies a suitable Montague type for the interpretation of the PTQ fragment along the lines of Part I. To narrow down our choice of single-type candidates, we first introduce a set of requirements which ensure the type's suitability as a single semantic basis for the PTQ fragment (in Sect. 4.1). The application of these requirements to the simplest Montague types (in Sect. 4.2) identifies the types $\langle s, t\rangle$ and $\langle s, \langle s, t\rangle\rangle$ as suitable single basic types. The chapter closes by comparing these two types to Partee's original single-type choice.

4.1. Single-Type Requirements

As one would expect, not all types which are obtained from the types e and $\langle s, t\rangle$ through the type-forming rule **CT** (hereafter called *Montague types*) are equally suitable as a single semantic basis for natural language. To identify the best (or most promising) single-type candidate, we require that this candidate satisfies the Properties (i) to (iv), below:

(i) Familiarity: *The single basic type figures in the formal semantic analysis of some linguistic phenomenon.*

(ii) Algebraicity: *The single-type domain has an algebraic structure.*

(iii) Representability: *All types of Montagovian objects can be represented via single-type objects.*

(iv) Simplicity: *Given its satisfaction of Properties (i) to (iii), the single basic type is obtained from the basic Montague types through the least number of* **CT***-applications.*

Property (i) ensures the proximity of our single-type semantics to mainstream theories of formal semantics. Property (ii) allows the interpretation of linguistic connectives as algebraic operations. Property (iii) enables the bootstrapping of representations of all Montagovian objects from objects of the single basic type. Property (iv) guarantees the low semantic complexity of single-type objects.

In virtue of their conceptual simplicity, the requirements from Properties (i) and (iv) do not demand further discussion. The requirements from Properties (ii)

and (iii) are discussed in Sections 4.1.1 and 4.1.2. The reader will note that the requirement from Property (ii) is semantically more general than the requirement from Property (iii). In fact, we will see in Section 4.2 that the property of representability is intimately connected to the semantics of a single basic type. Thus, our discussion of single-type requirements will give us a feel for what is necessary and what is possible in single-type semantics.

We start with a discussion of the requirement of algebraicity (Property (ii)).

4.1.1. Algebraicity. Algebraicity is the most general requirement on any single basic type. The obtaining of this property is demanded by the need to provide 'natural' interpretations of linguistic connectives (instead of the proxies from Ch. 2.1), and to give a formal basis for the relation of linguistic entailment. Many linguists have suggested that the English words and, or, and not act as algebraic operators between expressions of the same syntactic category.[1] But this interpretation of connectives is only possible if their domains have an algebraic structure. Further, since all single-type candidates are, by definition, the *only* basic type in their associated semantics, the algebraic structure of derived single-type domains depends entirely on the structure of the base domain. Thus, domains of a derived type only form an algebra if the set of basic-type objects forms an algebra.

We next present the representability requirement on single-type objects (Property (iii)).

4.1.2. Representability. The representability requirement on single-type candidates is a direct consequence of Partee's claim that it is possible to model the PTQ fragment through the use of a single basic type (Prop. 1.2). The requirement demands that objects of *any* suitable single basic type enable us to bootstrap representations of objects of *all* Montague types. The requirement of representability is expressed more formally below:

(iii)′ Representability: *Assume that a Montague type* α *is* related† *to some single-type type* β *if there exists, for every type-*α *object* a *exactly one type-*β *object* b *which represents* a*, s.t. the objects* a *and* b *are* one-to-one *related. Then, one of the following holds for every Montague type* α*:*

- (a) *The type* α *is* related† *to the single basic type o;*
- (b) *The type* α *is* related† *to some derived type* $(\beta_1 \dots \beta_n; o)$*, where* $\beta_1, \dots, \beta_n$ *are* (*basic or derived*) *single-type types.*

Given the existence of a unique single basic-type representation for every individual and for every proposition (cf. clause (a)), Church's type-forming rule **CT** en-

[1]Thus, the word and coordinates sentences (e.g. John sings and Mary dances), noun phrases (John and a unicorn), verb phrases (sings and dances), adjectival phrases (rhythmic and musical), prepositional phrases (on screen and in the real world), and many others.

sures the existence of a unique single-type representation for every Montagovian object of a derived type (cf. clause (b)). As a result, it suffices for a demonstration of the satisfaction of Property (iii) to show that (a) obtains. We will see concrete examples of the success and failure of the representability of single-type objects in Section 4.2.2, resp. 4.2.3. In Section 4.2.3 (cf. Prop. 4.1, 4.2), we show that our paradigmatic representable single-type candidates enable us to identify semantic equivalence relations between proper names and sentences.

Note that Property (iii) does *not* demand a bijective *correspondence* between individuals or propositions and their representations in the single basic type. Instead, it only demands an injective map between objects of each of these two types. This leaves open the existence of basic single-type objects which do *not* represent an individual or a proposition, and which represent *both* an individual *and* a proposition. As a result of the former, the function from individuals or propositions to their single-type representations may be non-surjective. As a result of the latter, the functions from the *union* of the sets of individuals and propositions to the set of basic single-type objects may be non-injective. We will see in Section 4.2.3 that the non-injectivity of the second function is a necessary condition for the accommodation of Proposition 1.3.ii.

The definition of representability from (iii)$'$ comprises two differently demanding requirements on single-type objects. These requirements are given below:

(iii.1) There exists a *sufficiently large set*, $\mathcal{O}$, of basic single-type objects;

(iii.2) There exists an *injective* map from Montagovian objects to their associated single-type objects.

In particular, we demand for the satisfaction of requirement (iii.1) that the cardinality, $|\mathcal{O}|$, of the basic single-type domain be *at least as large as* the cardinality, $|D_{\langle s,t\rangle}|$, of the set of Montagovian propositions (s.t. $|D_{\langle s,t\rangle}| \leq |\mathcal{O}|$). For the satisfaction of requirement (iii.2), we demand the specification of an injective function from individuals and from propositions to basic single-type objects. The first demand is motivated by our considerations from the previous paragraph, and by the common assumption that there are more propositions than individuals. The second demand is motivated by the impossibility of identifying the single-type representation of a Montagovian object in the absence of such a function. Since the satisfaction of the requirement from Property (iii.1) does not imply the satisfaction of the requirement from Property (iii.2), the large cardinality of the basic single-type domain is not a sufficient criterion for the satisfaction of Property (iii). We will return to the different representability requirements in Section 4.2.2.

This completes our discussion of the semantic requirements on the single basic type. We next show how these requirements can be used to identify a suitable single-type candidate (or candidates).

4.2. Identifying Suitable Candidates

Our introduction to this chapter has suggested a successive identification of suitable single-type candidates: From the set of Montague types, we pick out types on the basis of their ability to satisfy Properties (i) to (iv). Types which satisfy these properties are regarded as 'good' single-type candidates, which can serve as the foundation of our semantics.

We identify the domain of our search for witnesses of Properties (i)–(iii) with the set of the *simplest* Montague types, whose members are obtained through at most *two* applications of the type-forming rule **CT**. This restriction is justified by the existence of suitable single-type candidates in the resulting set (as we will see below), and by our adoption of the simplicity requirement on the single basic type from Property (iv). To identify the *simplest possible* single basic type, we replace the Montague types e and $\langle s,t\rangle$ by the types e, s, and t. This move corresponds to the adoption of a streamlined variant of Montague's type theory, which is due to Gallin (**1975**).

The application of the rule **CT** to the types e, s, and t yields the single-type candidates from Figure 4.1:

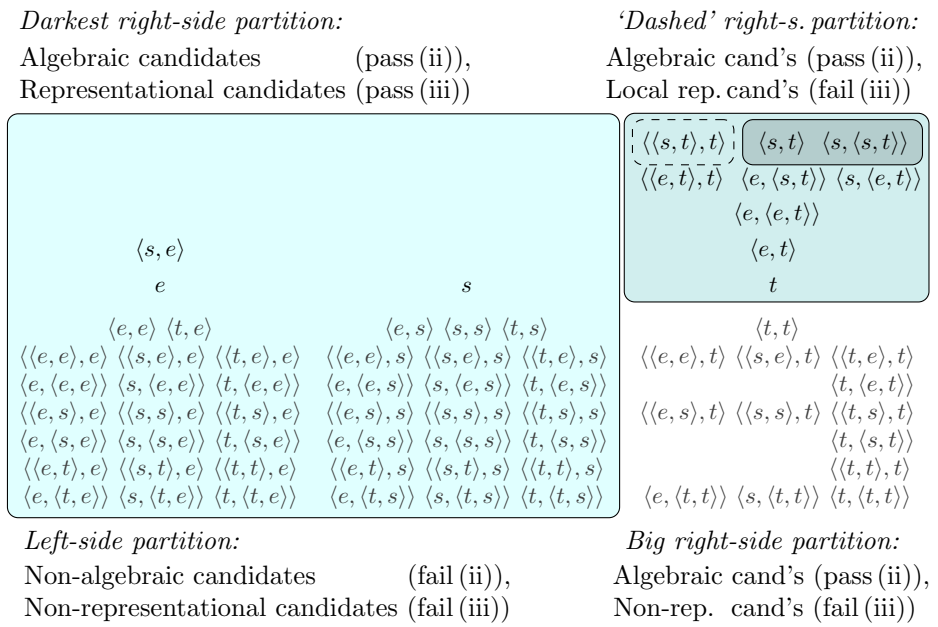

FIGURE 4.1. Single-type candidates and their suitability.

Sections 4.2.1 to 4.2.3 identify single-type candidates on the basis of their ability to satisfy Properties (ii) (cf. Sect. 4.2.1) and (iii) (cf. Sect. 4.2.2, 4.2.3). The decorations in Figure 4.1 summarize the reasons for the persistence or drop-out of each candidate. In the figure, Montague types which violate the requirement of familiarity (Property (i)) are printed in grey font.

4.2.1. Identifying Algebraic Types. The availability of an algebraic structure (cf. Property (ii)) constitutes one of the most effective criteria for the identification of potential single-type candidates in the set of types from Figure 4.1. This criterion relates to the greater ease of interpreting linguistic connectives in algebraic domains. In the introduction to this part, we have already suggested that the domain of the type t has an algebraic structure, such that all domains of some type $\langle\alpha_1, \langle \ldots, \langle\alpha_n, t\rangle\rangle\rangle$ inherit this structure through the lifting of operations on the set of truth-values. As a result, all candidates from the right-side partitions of Figure 4.1 are suitable single basic types from the point of view of Property (ii).

On the basis of Property (ii), candidates from the left-side partition of Figure 4.1 *dis*qualify as suitable single-type candidates. This is a result of the absence of an algebraic structure on the domains of individuals and indices in traditional Montague semantics, and the attendant non-algebraicity of all domains of some type $\langle\alpha_1, \langle \ldots, \langle\alpha_n, e\rangle\rangle\rangle$ or $\langle\alpha_1, \langle \ldots, \langle\alpha_n, s\rangle\rangle\rangle$. Since types of this form constitute two thirds of the types from Figure 4.1, the algebraicity requirement already enables us to exclude most of the candidate types as suitable single basic types.

Admittedly, the assumption of an algebraic structure on the domains of the types e and s (along the lines of (**Link, 1983**) and (**Kratzer, 1989**)) would prevent the fast exclusion of types $\langle\alpha_1, \langle \ldots, \langle\alpha_n, e\rangle\rangle\rangle$ and $\langle\alpha_1, \langle \ldots, \langle\alpha_n, s\rangle\rangle\rangle$ as suitable single-type candidates. As a result, individuals, indices, and individual concepts (type $\langle s, e\rangle$) would still qualify as single-type candidates on the basis of Property (ii). In the left partition of Figure 4.1, these types are printed in black font.

The idea of representing individuals and propositions in the type for indices, s, is supported by Kratzer's conception of situations as objects which contain individuals and facts, where facts are the truth-makers of propositions (**Kratzer, 1989**, pp. 612–613), cf. (**Partee, 2006**). The idea of representing individuals and propositions in the type for individuals, e, has its origins in Frege's characterization of truth-values as *Gegenstände* ['objects'], cf. (**Frege, 1891**). The latter representation has been proposed by some semanticists[2] as an obvious single-typing strategy. However, the required extension[3] of the set of individuals with the type-e correlates of some propositions violates our restriction to traditional Montague

[2]Proponents include Chierchia and Turner (**1988**), and Zoltán Gendler Szabó (p.c.).

[3]This extension is made necessary by the assumption that there are more propositions than individuals, and by the cardinality requirement on single-type domains (cf. Property (iii.1)).

(or Gallin) types – in particular, our adoption of Montague's type-e domain. Since the result of this extension would further be very similar to the semantics from Part I, an e-based single-type semantics will be unable to identify equivalence relations between proper names and sentences (cf. Prop. 3.3). An analogous argument prevents the adoption of a single-type semantics based on *indices*. We will argue in Section 4.2.2 that algebraic individual concepts constitute an equally unsuitable single basic type. On the example of individual concepts, this section will also contain a detailed illustration of the above problems.

This completes our identification of algebraic types in the set of single-type candidates from Figure 4.1. We next turn to the identification of representational algebraic types.

4.2.2. Excluding Non-Representational Types. The ability of *representing* Montagovian individuals and propositions is a more elusive criterion for the identification of suitable single-type candidates than algebraicity. This is due to the impossibility of inferring a type's satisfaction of Property (iii) from its superficial type structure. As a result, we need to check the representability of the remaining candidates from the big right-side partition in Figure 4.1 one-by-one. We do this by attempting to specify an injective function from individuals and from propositions to objects of the candidate type (cf. Sect. 4.1.2, (iii.2)): If the specification of such a function succeeds, we conclude the suitability of this type on the basis of Property (iii). If the specification of such a function proves difficult, we assume the type's potential[4] unsuitability on the basis of Property (iii).

Below, we will first consider a single-type candidate (i.e. the type t) whose domain contains too few objects for an injective representation of Montagovian individuals or propositions (s.t. this candidate does not satisfy Property (iii.1)). We will then consider a number of candidates (i.e. the types $\langle e,t\rangle$, $\langle e,\langle e,t\rangle\rangle$, $\langle s,\langle e,t\rangle\rangle$, $\langle e,\langle s,t\rangle\rangle$, $\langle\langle e,t\rangle,t\rangle$, and $\langle\langle s,t\rangle,t\rangle$) which lack a salient strategy for the injective representation of individuals or propositions (s.t. it is uncertain whether these candidates satisfy Property (iii.2)). In Section 4.2.3, we identify two candidates (i.e. the types $\langle s,t\rangle$ and $\langle s,\langle s,t\rangle\rangle$) whose salient strategy enables the representation of Montagovian individuals and propositions along the lines of Property (iii.2).

We start our elimination of non-(saliently) representational types with t:

Members of the domain of the type t (i.e. truth-values) do not enable the representation of Montagovian objects. This due to the small cardinality of the set of truth-values (i.e. 2) and to the comparatively large cardinality (i.e. $2^{|D_s|}$, with

[4]Notably, our *in*ability to identify an injective function from individuals or propositions to objects of the single basic type does *not* imply the non-existence of such a function: We can only infer the non-salience of this function. The presented argument for the identity of a suitable single basic type is thus only suggestive, not compelling.

$|D_s| > 2$) of the domain of propositions. As a result, the representation of propositions via truth-values will associate different objects of a particular Montague type with the same single-type object. But this violates our assumption of an injective map between the domains of these two types. Admittedly, the replacement of the set $\{\mathbf{T}, \mathbf{F}\}$ by a countably infinite set of truth-*degrees* would establish the desired relation. However, since there does not exist a principled relation between propositions and truth-degrees (along the lines described below), and since the interpretation of t in a domain other than $\{\mathbf{T}, \mathbf{F}\}$ violates our restriction to traditional Montague (or Gallin) types, the type t is ruled out as a suitable single basic type.

Since they do not allow an easy representation of basic Montagovian objects, extensional properties of individuals (type $\langle e, t\rangle$) and binary relations between individuals (type $\langle e, \langle e, t\rangle\rangle$) are also precluded from our further considerations. This is due to the fact that their salient representation strategy does not give an injective representation of propositions. Consider the case of extensional properties of individuals: Their adoption as basic single-type objects suggests the representation of individuals via their singleton sets. Since every individual a is bijectively related to the set $\{a\}$, this strategy satisfies the injectivity requirement for individuals. However, there does not exist a comparable strategy for the representation of propositions. This follows from the fact that few propositions attribute to their individual a uniquely identifying property (Fact 1) and that many propositions encode information about more than one individual (Fact 2).

As a consequence of Fact 1, the representation of propositions of the form Fa (where F denotes a property of individuals, type $\langle e, \langle s, t\rangle\rangle$) via the set in (4.2.1) may be ambiguous between the results of attributing the property F to any of the individuals which also have the property F at the current index @, such that the representation is not injective.

(4.2.1) $\{x_e \mid x \text{ has the property } F \text{ at } @\}$

For example, if, at @, the property 'walks' holds of John, Mary, and Bill, the type-$\langle e, t\rangle$ representation of 'Bill walks', i.e. $\{x_e \mid x \text{ walks at } @\} = \{\text{John}, \text{Mary}, \text{Bill}\}$, will be ambiguous between the representations of the propositions 'Bill walks', 'Mary walks', 'John walks', and their conjunctions and disjunctions. But this contradicts our assumption of a one-to-one mapping between propositions and basic single-type objects. Since most individuals will, at the current index, have more than one property, the extension of the restrictor on the set from (4.2.1) by the conjunct '$x = a$' (here, by '$x =$ Bill') (in (4.2.2)) does not rectify this situation.

(4.2.2) $\{x_e \mid x \text{ has the property } F \text{ at } @ \text{ and } x = a\}$

As a consequence of Fact 2, a single proposition can have different representations in the type $\langle e, t\rangle$, such that its representation is also not functional. For example, since the proposition 'John loves Mary' contains information about John

(namely, his loving Mary) as well as Mary (namely, her being loved by John), it can alternatively be represented by the sets $\{x_e \mid x \text{ loves Mary at } @\}$ or $\{x_e \mid x \text{ is loved by John at } @\}$. The representation of 'John loves Mary' via the *union* of the above sets, i.e. $\{x_e \mid x \text{ loves Mary or is loved by John at } @\}$, ensures the functionality of the representation relation on propositions. However, the result is, again, ambiguous between the representations of different propositions (e.g. the propositions 'John loves Mary', 'Someone loves Mary or is loved by John') and is, thus, non-injective. Since analogous observations hold for the representation of propositions in the type for binary relations between individuals, we exclude, next to the type $\langle e, t\rangle$, also the type $\langle e, \langle e, t\rangle\rangle$ from our further considerations.

The generalization of the representation strategy from (4.2.2) across indices (in (4.2.3)) – and the assumption that, for each pair of individuals and for each property, there exists an index at which only one of the individuals carries the property[5] – remedy the non-injectivity of the representations of propositions Fa in the type for *intensional* properties of individuals (or for functions from indices to extensional properties of individuals, type $\langle s, \langle e, t\rangle\rangle$). This is due to the possibility of restricting the carriers of the property F at every index to the individual a (if a has the property F at that index).

(4.2.3) $\quad \{\langle w_s, x_e\rangle \mid x \text{ has the property } F \text{ at } w \text{ and } x = a\}$

As a result, the characteristic function of the set from (4.2.3) will attribute to each pair, $\langle w, x\rangle$, of an index w and individual x the value **T** if x has the property F at w and x is a, and will attribute the value **F** if x lacks the property F at w or if x is not a. Let us return to the example from the previous page: Since we assume that, at some indices, Bill walks, but Mary does *not* walk, and that, at other indices, Bill walks, but *John* does not walk, the representation strategy from (4.2.3) excludes the representations of the propositions 'Mary walks' and 'John walks'.

Despite its merits, the representation strategy from (4.2.3) still fails to ensure the functional representation of relational propositions like 'John loves Mary'. Of course, we can restrict the second element of the ordered pairs in the set $\{\langle w_s, x_e\rangle \mid x \text{ loves Mary or is loved by John at } w\}$ to John or Mary. However, since the result remains ambiguous between the representations of 'John loves Mary', 'Mary loves herself or is loved by John', etc., the strategy from (4.2.3) is not suitable for the representation of propositions in the type $\langle s, \langle e, t\rangle\rangle$.

Note that the unsuitability of the representational strategy from (4.2.3) does not imply the *general* unsuitability of the type $\langle s, \langle e, t\rangle\rangle$ as single-type candidate: For example, the representation of propositions φ via the characteristic function of the set from (4.2.4) satisfies the injectivity requirement from Property (iii.2).

(4.2.4) $\quad \{\langle w_s, x_e\rangle \mid w \in \varphi\}$

[5]The strength of this assumption is another reason for the rejection of the presented strategy.

However, since the less complex type $\langle s, t\rangle$ enables an equivalent representation of propositions (as we will see in Sect. 4.2.3, cf. (4.2.9)), the type $\langle s, \langle e, t\rangle\rangle$ (and, analogously, the type $\langle e, \langle s, t\rangle\rangle$) does not satisfy the requirement of simplicity from Property (iv). As a result, the types $\langle s, \langle e, t\rangle\rangle$ and $\langle e, \langle s, t\rangle\rangle$ are also excluded from our further considerations. A similar observation can be made with respect to the type for generalized quantifiers over indices, $\langle\langle s, t\rangle, t\rangle$.

Our discussion of the types $\langle e, t\rangle$ and $\langle s, \langle e, t\rangle\rangle$ from the last two pages has suggested the possibility of representing Montagovian individuals and propositions via *sets* of type-$\langle e, t\rangle$ objects (i.e. via generalized quantifiers over individuals, type $\langle\langle e, t\rangle, t\rangle$). Every individual a can then be represented via the set of its extensional properties (in (4.2.5)):

$$\{P_{\langle e,t\rangle} \mid a \in P\} \tag{4.2.5}$$

$$\{P_{\langle e,t\rangle} \mid a \in P\} \cup \big\{\{x_e \mid x \text{ has the property } F \text{ at } @\}\big\} \tag{4.2.6}$$

The representation of individuals from (4.2.5) suggests the possibility of representing propositions via the *union* of the type-$\langle\langle e, t\rangle, t\rangle$ representation of the propositions' type-e argument(s) and the singleton containing the arguments' attributed property. In (4.2.6), this strategy is formulated for propositions of the form Fa. Yet, since the result has similar defects as the representation of propositions in the types $\langle e, t\rangle$ and $\langle s, \langle e, t\rangle\rangle$, we drop the type $\langle\langle e, t\rangle, t\rangle$ from our considerations.

We finish our exclusion of non-saliently representing single-type candidates by showing the unsuitability of the type $\langle s, e\rangle$ as a single basic type.

Our assumption of a hypothetical algebraic structure on the domain of individuals (Sect. 4.2.1) has suggested the possibility of interpreting the PTQ fragment in the type for individual concepts, $\langle s, e\rangle$. The adequacy of the type $\langle s, e\rangle$ as a single basic type is supported by its comparative simplicity (cf. Property (iv)), by the prominence of this type in the ontology from (**Montague, 1973**) (cf. Property (i)), and by the greater cardinality (i.e. $|D_e|^{|D_s|}$) of the set of individual concepts with respect to the cardinality (i.e. $2^{|D_s|}$) of the set of propositions (cf. Property (iii.1)).

These desirable properties notwithstanding, we will disregard the type $\langle s, e\rangle$ as a single basic type. This is due to the impossibility of predicting a name's intuitively equivalent sentences in a given context (cf. Prop. 1.3.ii). To represent Montagovian objects in the type for individual concepts, one could extend the set of individuals via the type-e correlates of **T** and **F** (cf. Sect. 4.2.1), represent every individual a by a constant function from indices to a (cf. the graph in (4.2.7)), and represent every proposition φ by a function from indices w to the individual correlate of φ's truth-value at w, $\varphi(w)$ (cf. the graph in (4.2.8), next page):

$$\big\{\langle w_s, x_e\rangle \mid x = a\big\} \tag{4.2.7}$$

(4.2.8) $\{\langle w_s, x_e\rangle \mid x \text{ is the type-}e \text{ correlate of } \varphi(w)\}$

However, as we have pointed out in Section 4.2.1, an extension of the set of individuals with correlates of the two truth-values violates our restriction to *Montagovian* individuals. Since the representation of propositions as functions to *non-Montagovian* individuals renders the function from the union of individuals and propositions to their single-type representations *injective* (s.t. no individual concept represents both an individual and a proposition), it further prevents the identification of equivalence relations between names and sentences (cf. Prop. 3.3).

To avoid these problems, one could identify the type-e correlates of the truth-values **T** and **F** instead with different *Montagovian* individuals. However, besides the difficulty of determining which individuals should double as truth-values, the resulting single-type semantics would make counterintuitive predictions about the above-mentioned equivalence relations.

Consider the type-$\langle s, e\rangle$ representation of the proposition 'John runs' at an index, @, which is inhabited by three individuals: John, Mary, and Bill. Assume that, at @, the proposition 'John runs' is true. Assume further that Mary serves as the individual correlate of **T**, and that Bill serves as the individual correlate of **F**. Then, since 'John runs' is evaluated '**T**' at @, the type-$\langle s, e\rangle$ representation of 'John runs' will send @ to Mary. Since the pair $\langle$@, Mary$\rangle$ is further a member of the type-$\langle s, e\rangle$ representation of Mary (by the strategy from (4.2.7)), the sentence John runs is semantically equivalent to the proper name Mary at @ in this semantics. But this goes against the intuitions described in Chapter 1.2.1.

The interpretation of proper names as *partial* functions from indices to individuals (which fail to output an individual at some indices) may, in some cases, prevent the derivation of the above-asserted equivalences: If it so happens that, in the situation from the previous paragraph, Mary does not exist in @, we will not be able to interpret John runs as the semantic value of Mary at @ in our $\langle s, e\rangle$-based single-type semantics. However, since the equivalence of two expressions at an index cannot, in principle, be excluded, the partialized representation strategy is also unsuitable for the representation of individuals and propositions in the type $\langle s, e\rangle$. As a result, we also exclude the type $\langle s, e\rangle$ from our further considerations.

This completes our excursus on the possibility of an $\langle s, e\rangle$-based single-type semantics. We next show the representability of the remaining single-type candidates from Figure 4.1.

4.2.3. Identifying Representational Types. Since the types $\langle s, t\rangle$ and $\langle s, \langle s, t\rangle\rangle$ enable the injective representation of Montagovian individuals and propositions (see below), they both constitute promising single-type candidates. We begin by describing the salient representational strategy of the type $\langle s, t\rangle$.

Our discussion of the type $\langle s, \langle e, t\rangle\rangle$ from the previous section has already indicated the possibility of representing individuals and propositions in the type for functions from indices to truth-values, $\langle s, t\rangle$. This type enables the representation of every individual a by the characteristic function of the set of all indices in which this individual exists, and enables the representation of every proposition φ by the characteristic function of the set of all indices at which the proposition's designator is true. In a single-type semantics based on the type $\langle s, t\rangle$, propositions will, thus, be represented by 'themselves' (cf. the representation of φ in (4.2.9), below).

In the type-$\langle s, t\rangle$ representation of individuals (in (4.2.10)), an individual's *existence in* an index is understood as the individual's '*being in* an index' (i.e. as the individual's *inhabitance of* that index). This understanding of existence corresponds to our pre-theoretical intuitions about existence, and to the understanding of *concreteness* (i.e. the occupancy of a spatio-temporal position) in fixed-domain quantified modal logic, cf. (**Linsky and Zalta, 1994**). As a result of the former, an individual either exists or does not exist in each index. As a result of the latter, the same individual can exist in different indices (i.e. individuals can exist *across* indices). In Chapter 6.1, the described behavior of existence will be captured in the axioms **Ax10** and **Ax11**, respectively.

(4.2.9) $\{w_s \mid w \in \varphi\}$

(4.2.10) $\{w_s \mid a \text{ exists in } w\}$

To ensure the injectivity of the individual-representations from (4.2.10) (s.t. no two individuals are represented by the same set of indices), we postulate that, for every pair of individuals, there is an index at which one, but not the other individual exists. This assumption – which will be captured in axiom **Ax12** – is common in Situation Semantics, cf. (**Muskens, 1995**, pp. 70–71), and in the semantics of quantified modal logic. It will receive further justification and discussion in Chapter 6.1.

Since the characteristic functions of the sets from (4.2.9) and (4.2.10) represent every individual a in the same single-type object as the proposition 'a exists', their union is non-injective. This observation motivates the following claim:

Proposition 4.1 (Weak Single-Type Hypothesis). *In an $\langle s, t\rangle$-based single-type semantics, the representations of individuals are identical to the result of attributing them an existence property.*

In virtue of the strategy from (4.2.10), the representations of individuals in the type $\langle s, t\rangle$ carry only very weak semantic information, such that they cannot encode information about an individual's contextually salient properties besides existence. By itself, the informational poverty of type-$\langle s, t\rangle$ representations of individuals and propositions is unproblematic. However, we will see in Part III that an $\langle s, t\rangle$-based single-type semantics fails to identify a name's contingent senten-

tial equivalents (cf. Prop. 1.3.ii).

To accommodate Proposition 1.3.ii, we provide a semantically richer representation of individuals and propositions. This representation proceeds in the type for functions from indices to propositions, $\langle s, \langle s, t\rangle\rangle$. In analogy with the name for type-$\langle s, e\rangle$ objects (which are called *individual concepts*), we refer to functions of the type $\langle s, \langle s, t\rangle\rangle$ as *propositional concepts*.[6]

The particular representation strategy of propositional concepts follows the strategy for the representation of individuals and propositions in the type $\langle\langle e, t\rangle, t\rangle$ (cf. (4.2.5), (4.2.6)). Only, rather than representing every individual a via the set of its extensional properties at the current index @, we represent this individual via the set of indices at which all true propositions at @ which carry information about the individual are true. To facilitate the introduction of this new representation strategy, we first consider representations of individuals *at the current index*. These representations proceed in the familiar type for Montagovian propositions, $\langle s, t\rangle$. Propositional concepts (type $\langle \underline{s}, \langle s, t\rangle\rangle$) will only be required for the generalization of this representation strategy *across* indices.

At the index @, every individual a is represented by the characteristic function of the set of indices from (4.2.11) (below). In the restrictor of this set, the *aboutness* of a proposition p with respect to an individual a is defined as follows:

Definition 4.2.1 (Aboutness). A proposition p *is about* the individual a iff p is equivalent to a formula of the form Fa (with F a type-$\langle e, \langle s, t\rangle\rangle$ constant).

For example, since the logical correlates, Rj and $\overline{Rm}$, of the sentences John runs and Mary does not run contain the correlates, i.e. j and m, of the names John and Mary as constituents, the propositions 'John runs' and 'Mary does not run' will be about John, resp. Mary. Similarly, since the correlates of the sentences John loves Mary and John runs and Mary does not run both contain the correlates of the names John *and* Mary, the propositions denoted by these sentences will both be about John *and* Mary. We will see in Chapter 6.1 that the behavior of the aboutness predicate can be derived from the small set of axioms from (**Perry, 1986**, p. 129).

$$\{w_s \mid \text{for all } p_{\langle s,t\rangle}, \text{ if } @ \in p \text{ and } p \text{ is about } a, \text{ then } w \in p\} \tag{4.2.11}$$

Axiom **Ax12** ensures the injectivity of type-$\langle s, t\rangle$ individual-representations: Since there is, for every pair of individuals, an index at which one, but not the other individual exists, no two individuals will, at @, be represented by the same set of indices. This even holds of individuals which have *exactly the same properties at* @ (including the property 'exists in @'): For two such individuals b and c, the aboutness-relevant @-true propositions 'b exists' (resp. 'does not exist') and 'c exists' (resp. 'does not exist') will, by **Ax12**, never be true at the same indices.

[6]This name has been suggested by Jeroen Groenendijk.

For brevity, we will hereafter denote the type-$\langle s, \langle s, t\rangle\rangle$ representation of an individual a at the index @ from (4.2.11) by '$a^{\dagger}_{@}$'.

To aid the reader's understanding of our strategy for the type-$\langle s, \langle s, t\rangle\rangle$ representation of individuals, we illustrate this strategy by means of an example: Consider the representation, $j^{\dagger}_{@}$, of John at @ in a universe consisting of three indices, @, w_1, and w_2, and two individuals: John (abbreviated j) and Mary (m). Assume that Mary exists in all indices, s.t. the proposition 'Mary exists' (Em) is true at @, w_1, and w_2, and that John exists in the indices @ and w_1, s.t. the proposition 'John exists' (Ej) is true at @ and w_1.[7] Assume further that, at the current index, the propositions 'John runs' (Rj) and 'Mary runs' (Rm) are true (s.t. Ej, Em, Rj, and Rm obtain at @), that, at the index w_1, the propositions 'John runs' and 'Mary doesn't run' ($\overline{Rm}$) are true (s.t. Ej, Em, Rj, and $\overline{Rm}$ obtain at w_1), and that, at the index w_2, the propositions 'John doesn't run' and 'Mary runs' are true (s.t. $\overline{Ej}$, Em, $\overline{Rj}$, and Rm obtain at w_2) (cf. Fig. 4.2).

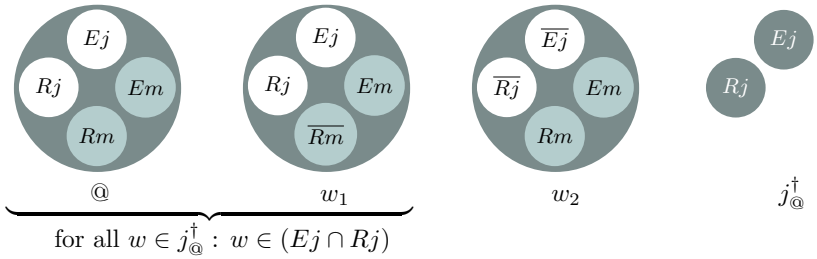

FIGURE 4.2. An $\langle s, \langle s, t\rangle\rangle$-representation of John at @.

Then, by our definition of aboutness, 'John exists' and 'John runs' are the only true propositions at @ which carry information about John. As a result, we represent John at @ by the subset of the set $\{@, w_1, w_2\}$ at whose members John exists and runs. We identify this subset with the set $\{@, w_1\}$ (underbraced in Fig. 4.2), which encodes the information that John, who exists, runs.

To enable the type-$\langle s, \langle s, t\rangle\rangle$ representation of individuals at other indices – and, thus, to enable the identification of equivalence relations between proper names and sentences at those indices (cf. Prop. 1.3.ii) –, we generalize the representation strategy from (4.2.11) to the strategy from (4.2.12):

(4.2.12) $\big\{\langle w_1, w\rangle \mid$ for all $p_{\langle s,t\rangle}$, if $w_1 \in p$ and p is about a, then $w \in p\big\}$

[7] The non-existence of John in w_2 (s.t. $\{w_s \mid$ John exists in $w\} \neq \{w_s \mid$ Mary exists in $w\}$) witnesses the requirement from **Ax12**.

The above is equivalent to a function from indices to the rich type-$\langle s, t\rangle$ representation of the individual a at those indices. In line with our earlier conventions, we will sometimes denote the object from (4.2.12) by '$a^{\dagger}$'.

Objects of the described form are familiar from intensional, indexical, and dynamic semantics. Thus, in Landman's Data Semantics, cf. (**Landman, 1984**), individuals in an information state σ are represented via the sets of true propositions at σ which carry information about them. In Kaplan's semantics for indexical expressions, cf. (**Kaplan, 1989**), *character* is described as a function from situational contexts (type s) to semantic contents (type e, or $\langle s, t\rangle$). In Muskens' (**1996**) type-theoretic formulation of Discourse Representation Theory, discourse representation structures are treated as functions from states to propositions.

On the basis of the above, we next turn to the type-$\langle s, \langle s, t\rangle\rangle$ representation of *propositions*.

The most straightforward strategy for the representation of propositions lies in the identification of propositions with (the characteristic function of) the set of all indices at which they are true (cf. (4.2.9)). This set can be lifted to the type for propositional concepts by taking the constant function from indices to this set:

$$\{\langle w_1, w\rangle \mid w \in \varphi\} \tag{4.2.13}$$

However, representations of the form from (4.2.13) violate our intuition that the single-type representations of propositions are semantically at least as rich as the representations of the individuals about which they carry information (hereafter, the propositions' *aboutness subjects*). As a result, the representation of propositions by objects of the form from (4.2.13) blocks the identification of the representations of contextually salient propositions with the representations of their aboutness subjects (Prop. 1.3.ii).

To accommodate a semantically richer representation of propositions, we represent the truth-value of every proposition φ at the current index @ via the intersection of the set of indices at which φ is true (cf. (4.2.9)) and (the intersection of) the type-$\langle s, \langle s, t\rangle\rangle$ representation(s) of φ's aboutness subject(s) at @ (cf. (4.2.11)):

$$\begin{aligned} &\{w_s \mid w \in \varphi\} \cap \Big(\bigcap_{\varphi \text{ is about } x_e} x^{\dagger}_{@} \Big) \\ = \ &\{w_s \mid w \in \varphi \text{ and, for all } p_{\langle s,t\rangle}, \text{ if } @ \in p \text{ and,} \\ &\qquad \text{for some } x_e, \varphi \text{ is about } x \text{ and } p \text{ is about } x, \text{ then } w \in p\} \end{aligned} \tag{4.2.14}$$

We will sometimes denote the resulting set by '$\varphi^{\dagger}_{@}$'.

We also illustrate our strategy for the type-$\langle s, \langle s, t\rangle\rangle$ representation of propositions by means of an example: Consider the representation of the proposition 'John loves Mary' (Lmj) at @ in a universe consisting of three indices, @, w_1, and

w_2, and three individuals, John (j), Mary (m), and Bill (b). Assume that Mary exists in all indices, that John exists in @ and w_1, and that Bill only exists in @. Assume further that, at the current index and at the index w_1, the proposition 'John loves Mary' is true (s.t. Ej, Em, Eb, and Lmj obtain at @, and Ej, Em, $\overline{Eb}$, and Lmj obtain at w_1) and that, at the index w_2, the proposition 'John loves Mary' is false (s.t. $\overline{Ej}$, Em, $\overline{Eb}$, and $\overline{Lmj}$ obtain at w_2) (cf. Fig. 4.3).

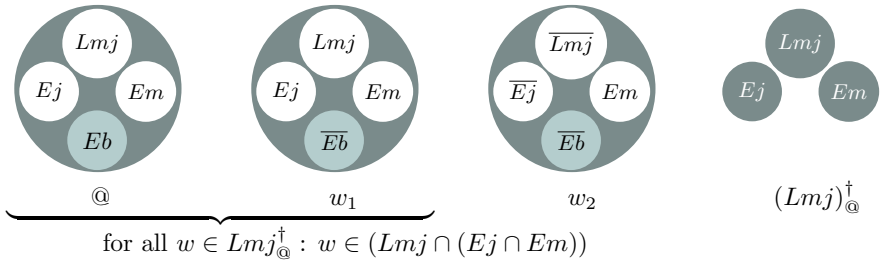

FIGURE 4.3. An $\langle s, \langle s, t\rangle\rangle$-representation of 'John loves Mary' at @.

Then, 'John loves Mary', 'John exists', and 'Mary exists' are the only true propositions at @ which carry information about the aboutness subjects (i.e. John and Mary) of the proposition 'John loves Mary'. As a result, we represent the truth-value of the proposition 'John loves Mary' at @ by the subset of the set $\{@, w_1, w_2\}$ at whose members the propositions 'John loves Mary', 'John exists', and 'Mary exists' are true. This subset is the set $\{@, w_1\}$ (underbraced in Fig. 4.3).

The representation strategies from (4.2.11) and (4.2.14) lead us to expect the following for an $\langle s, \langle s, t\rangle\rangle$-based single-type semantics:

PROPOSITION 4.2 (Strong Single-Type Hypothesis). *If a proposition is true at a given index w, its type-$\langle s, \langle s, t\rangle\rangle$ representation at w is equivalent to (the intersection of) the representations of the proposition's aboutness subject(s) at w.*

For example, since the proposition 'John loves Mary' is true at the index w_1 from Figure 4.3, its type-$\langle s, \langle s, t\rangle\rangle$ representation at w_1 (in (4.2.15)) is exactly the intersection (in (4.2.16)) of the type-$\langle s, \langle s, t\rangle\rangle$ representations of its aboutness subjects, John and Mary, at w_1:

(4.2.15) $$\begin{aligned} Lmj^{\dagger}_{w_1} &= \{w_s \mid \text{John loves Mary at } w\} \cap (j^{\dagger}_{w_1} \cap m^{\dagger}_{w_1}) \\ &= \{w_s \mid \text{John loves Mary at } w\} \cap \\ &\quad \{w_s \mid \text{John exists and loves Mary at } w\} \cap \\ &\quad \{w_s \mid \text{Mary exists and is loved by John at } w\} \end{aligned}$$

(4.2.16) $$= (j^{\dagger}_{w_1} \cap m^{\dagger}_{w_1})$$

Note that, if a proposition is *false* at the current index, the intersection of its index set with the representation of its aboutness subjects at @ will be empty. As a result, the representation of the proposition φ at @ will be identified with the representation of all propositions which fail to be true at @. To ensure that such cases do not disable the injective representation of propositions, we generalize the representation strategy from (4.2.14) to the strategy from (4.2.17):

(4.2.17) $\big\{\langle w_1, w\rangle \mid w \in \varphi$ and, for all $p_{\langle s,t\rangle}$, if $w_1 \in p$ and, for some x, φ is about x and p is about x, then $w \in p\big\}$

We will sometimes denote objects of the above form by '$\varphi^{\dagger}$'.

This completes our identification of single-type candidates on the basis of their salient satisfaction of Property (iii). We close the chapter by discussing one further requirement on suitable single basic types.

4.2.4. Partializing the Suitable Types. Our previous considerations have presupposed a traditional Montagovian interpretation of indices as possible worlds (cf. Sect. 1.1.1). Since possible worlds are totally defined, we assume that the designator of every proposition p will be either true or false at each world w (s.t. $w \in p$ or $w \in \neg p$). However, the identification of indices with possible worlds yields undesirable consequences for the single-type semantics of the types $\langle s, t\rangle$ and $\langle s, \langle s, t\rangle\rangle$.

In particular, since we have identified an individual's type-$\langle s, \langle s, t\rangle\rangle$ representation at an index w with the representations of *all* true propositions at w which carry information about the individual (cf. Prop. 4.2), every name a will, at each index, be equivalent to the result of merging the name with the designator, F (or $\neg F$), of (the complement of) every identifiable property: If $w \in Fa$, then Fa will be judged equivalent to a; if $w \in \neg Fa$, then $\neg Fa$ will be judged equivalent to a in our $\langle s, \langle s, t\rangle\rangle$-based semantics. But this contradicts the empirical findings from Chapter 1.2.1, where the sets of admissible name-equivalents are typically *very small*.

For instance, in Chapter 1.2.1, only (equivalents of) the sentences Barbara Partee is entering the room (cf. (9b)) and Barbara Partee is arriving (cf. (9c)) are judged to be equivalent to the name Barbara Partee from (9a). However, if it is also true (or false) at the relevant index[8] that Barbara Partee is thinking about semantics, our representation strategies from (4.2.12) and (4.2.17) also predict the equivalence of the name Barbara Partee with (the negation of) the sentence Barbara Partee is thinking about semantics. But this violates our intuitions about the equivalence relations between proper names and sentences from Chapter 1.2.1.

[8]The relevant index is here understood as a possible world w of which the descriptions from (9) are true (s.t. Barbara Partee is arriving at w by entering the room).

The intuitive *non*-equivalence of the above expressions is illustrated by the possibility (in (22c), below) of negating the names' predicted sentential equivalents without the loss of consistency. This possibility suggests that the semantic information encoded in the sentence Barbara Partee is thinking about semantics is *not* contained in the information of the name Barbara Partee:

(22) CONTEXT: A woman is entering the room. A linguist turns to her friend, gestures towards the door, and says (a).

a. [$_{\text{NP}}$Barbara Partee]

b. * [$_{\text{NP}}$Barbara Partee] is *not* entering the room.

c. [$_{\text{NP}}$Barbara Partee] is *not* thinking about semantics.

Since – granted the context from (9) – the negation of (9b) (in (22b)) generates a contradiction, it suggests that the semantic information encoded in the sentence Barbara Partee is entering the room (and, similarly, for the sentence Barbara Partee is arriving) *is* contained in the information of the name Barbara Partee.

Below, we will first explain our intuitions behind the restrictions on the equivalence set of the name Barbara Partee from (9a). We will then present a strategy for the accommodation of these intuitions in an $\langle s, \langle s, t\rangle\rangle$-based semantics. We will see that this strategy also explains the small size of the equivalence sets in the other examples from Chapter 1.2.1.

In (9), the small size of the equivalence set of the name Barbara Partee is justified by the name's context of use. We identify this context with a *part* of the current index (called a *partial* possible world, or a possible *situation*; cf. (**Barwise and Perry, 1983**; **Kratzer, 1989**)). Partial possible worlds only determine the truth or falsity of *a proper subset* of the set of true or false propositions at @. As a result, the designators of some propositions p will be neither true or false at a situation w (s.t. $w \notin p$ and $w \notin \neg p$).

Common grounds (**Stalnaker, 1978**), visual scenes (**Barwise, 1981**), and information (or knowledge) states (**Veltman, 1996**) are good candidates for partial possible worlds. For example, in the situation from (9), the relevant partial world can be identified with the visual scene of Barbara Partee entering the room. We assume that the two participants of the situation are presented this scene (s.t. they both witness Partee's entering the room at approximately the same time). Then, since (equivalents and entailments of) the sentence Barbara Partee is entering the room are the only true sentences about Barbara Partee at the shared scene, the individual Barbara Partee will be represented via the proposition 'Barbara Partee is entering the room' in an $\langle s, \langle s, t\rangle\rangle$-based single-type semantics. As a result, (equivalents of) the sentence Barbara Partee is entering the room will be the only sentential equivalents of the name Barbara Partee at the presented scene in an $\langle s, \langle s, t\rangle\rangle$-based single-type semantics.

Notably, the presented strategy for the restriction of a name's sentential equivalents is only applicable if the designators of some propositions p are neither true nor false at some situations w (s.t. $w \notin p$ and $w \notin \neg p$). But this is only possible if we extend the set of truth-values, $\{\mathbf{T}, \mathbf{F}\}$, via the undefined truth-value $\mathbf{N}$ (*neither-true-nor-false*). Since we can represent the resulting elements *true*, *false*, and *neither-true-nor-false* by the sets of truth-values $\{\mathbf{T}\}$, $\{\mathbf{F}\}$, and $\emptyset$, respectively, we will hereafter refer to these elements as *truth-combinations*. The possibility of evaluating a sentence as '$\mathbf{N}$' at a situation enables the exclusion of this sentence from the set of true (or false) sentences at this situation (s.t. the truth or falsity of this sentence is not decided by the situation). By our strategy for the representation of individuals from (4.2.12), this sentence will then disqualify as the sentential equivalent of any proper name at that situation. For example, since the sentence Barbara Partee is thinking about semantics receives the truth-combination '$\mathbf{N}$' at the visual scene from (9), it does not qualify as a semantic equivalent of the name Barbara Partee at this scene.

Beyond the above, the partialization of the set of truth-values is motivated by our wish to preserve the standard behavior of negation in the type-$\langle s, t\rangle$ representation of individuals. Conservative semantics, cf. (**Russell, 1905**), evaluate both the result, Fa, and the negation, $\neg Fa$, of the result of attributing a contextually salient property F to an individual a at an index w where a does not exist as '$\mathbf{F}$'. For example, since Vulcan does not exist in the actual world, such semantics evaluate both the sentence Vulcan is a planet and the sentence It is not the case that Vulcan is a planet (or Vulcan is not a planet) as *false*. However, this violates the familiar axioms for negation.[9] Since the truth-combination $\mathbf{N}$ is uncomplemented (s.t. $-\mathbf{N} = \mathbf{N}$), the evaluation of both Fa and $\neg Fa$ at w as '$\mathbf{N}$', cf. (**Strawson, 1950**), preserves the familiar behavior of negation.

This completes the presentation of our strategy for the restriction of name/sentence-equivalences to intuitively admissible equivalences. We close the section by justifying our partialization of type-s and -t objects, and by adapting our representation strategies from (4.2.12) and (4.2.17) to these partial objects.

Our generalization of worlds and truth-values to *situations* and truth-*combinations* is justified by the fact that functions from situations to truth-combinations (type $\langle s, t\rangle$) are *equivalent* (*up to coding*) to functions from (functions from worlds to truth-values) (i.e. from *total* sets of worlds) to functions from truth-values to truth-values (i.e. to *total* sets of truth-values) (type $\langle\langle s, t\rangle, \langle t, t\rangle\rangle$), and by the fact that functions from situations to (functions from situations to truth-combinations) (i.e. functions to *partial* sets of situations; type $\langle s, \langle s,\, t\rangle\rangle$) are *equivalent* to functions from total sets of worlds to functions from worlds to total sets of truth-values (i.e. to *partial* sets of worlds) (type $\langle\langle s, t\rangle, \langle s, \langle t, t\rangle\rangle\rangle$).

[9] According to the axiom of Top and Bottom (cf. **Ax9**), if $Fa(w) = \mathbf{F}$, then $\neg Fa(w) = \mathbf{T}$.

The possibility of coding situations as total sets of worlds[10] is enabled by a corollary[11] of Stone's Theorem (**Stone, 1936**), cf. (**Davey and Priestley, 2002**). For our present purposes, this corollary is interpreted as stating that every situation w can be coded by the set of all worlds w_1 which maximally extend the semantic information encoded in w. This is the case if the designators of all propositions p which receive a classical truth-value at w (s.t. $p(w) = \mathbf{T}$ or $p(w) = \mathbf{F}$) preserve their truth-value at w_1. Total sets of worlds of the above form (called *possibilities*) are a standard tool in formal semantics and epistemic logic, cf. (**Groenendijk and Stokhof, 1991**; **Groenendijk and Roelofsen, 2009**).

The possibility of coding truth-combinations as total sets of truth-values[12] is enabled by the fact that the domain of the type $\langle t, t\rangle$ has four total functions as its members. The descriptions and the characteristic sets of these functions are given below:

(i)	the function which outputs its input:	$\{\mathbf{T}\}$
(ii)	the function which outputs the complement of its input:	$\{\mathbf{F}\}$
(iii)	the function which outputs **F** for every input:	$\emptyset$
(iv)	the function which outputs **T** for every input:	$\{\mathbf{T}, \mathbf{F}\}$

Since the characteristic sets of the functions from (i) to (iii) correspond to the truth-combinations *true-and-not-false*, *false-and-not-true*, and *neither-true-nor-false*, respectively, they provide a 'natural' coding of these combinations. Since our representation strategy does not require a fourth truth-combination (typically, *both-true-and-false*, '**B**'), we neglect the function from (iv).

The possibility of coding the types for functions from situations to truth-combinations and from situations to (functions from situations to truth-combinations) in terms of traditional Montague (or Gallin) types justifies our association of the types s and t with sets of possible *situations* and truth-*combinations*, which are not traditionally used in Montague semantics. To remind the reader of our partial interpretation of these types, we will hereafter sometimes refer to the types s and t as *partial* types.

As a result of our replacement of total by *partial* Gallin types, it may happen that the designator of a proposition φ may be neither true nor false at the representing situation w_1. To compensate for such occurrences, we require that φ be true at every member of its type-$\langle s, \langle s, t\rangle\rangle$ representation at w_1 (cf. the first con-

[10]This possibility regards the coding of the underlined types on the left of the arrows from $\langle \underline{s}, t\rangle \hookrightarrow \langle \underline{\langle s, t\rangle}, \langle t, t\rangle\rangle$ and $\langle \underline{s}, \langle s, t\rangle\rangle \hookrightarrow \langle \underline{\langle s, t\rangle}, \langle s, t\rangle\rangle$ to the underlined types on the right of the arrows.

[11]This corollary states that every filter in a distributive lattice is the intersection of all prime filters extending that filter.

[12]This possibility regards the coding of the underlined types left of the arrows from $\langle s, \underline{t}\rangle \hookrightarrow \langle \langle s, t\rangle, \underline{\langle t, t\rangle}\rangle$ and $\langle s, \langle s, \underline{t}\rangle\rangle \hookrightarrow \langle \langle s, t\rangle, \langle s, \underline{\langle t, t\rangle}\rangle\rangle$ to the underlined types on the right of the arrows.

junct in the representation scheme from (4.2.14)). Thus, (4.2.14) expresses an update on the available information at w_1 via the information encoded in the proposition φ.

Our replacement of total by partial Gallin types even requires an *adaptation* of the representational strategies of the type $\langle s, \langle s, t\rangle\rangle$ from (4.2.12) and (4.2.17). This is due to the fact that we associate the type $\langle s, t\rangle$ with *partial* sets of *situations*. In Section 4.2.3, we have specified that the restrictor on the set of ordered pairs from (4.2.12) (i.e. 'for all p, if $w_1 \in p$ and p is about a, then $w \in p$') will be satisfied by all indices w where all true propositions at w_1 which carry information about a are true, and whose false propositions either fail to be true at w_1 or fail to carry information about a. But this excludes the possibility of representing an individual at a partial situation via a proper extension of this situation which contains 'new' information about the represented individual. This is the case if an aboutness-relevant proposition whose truth-value is undefined at the original situation receives a classical truth-value at the representing situation.

Consider the following partial variant[13] (in Fig. 4.4) of the example from Figure 4.2. In particular, we now consider the representation of John at the partial *situation* @ in a universe consisting of the *situations* @, w_2, and w_3, and the individuals John (j) and Mary (m). We again assume that Mary exists in all situations, and that John exists in the situations @ and w_3. We further assume that, at the situation w_3, the proposition 'John runs' is true, and that, at the situation w_2, the proposition 'Mary runs' is true. As a result, Ej and Em obtain at @, Ej, Em, and Rj obtain at w_3, and $\overline{Ej}$, Em, and Rm obtain at w_2:

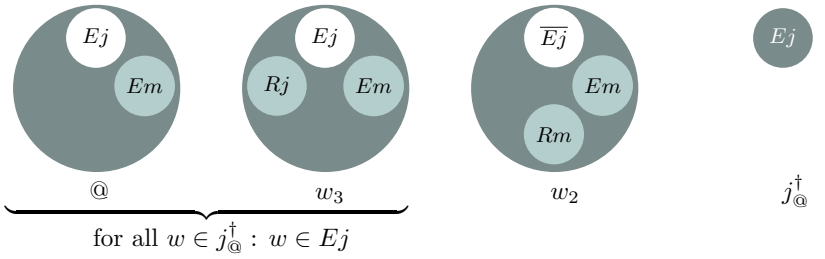

FIGURE 4.4. An $\langle s, \langle s, t\rangle\rangle$-representation of John at partial @.

Since the proposition 'John exists' is the only true proposition at @ which carries information about John, we expect that, at @, John will be represented by the set of the situations @ and w_3 (underbraced in Fig. 4.4). However, this set does not satisfy the representation strategy from (4.2.12). In particular, in the example

[13]To avoid a confusion of variables, we have changed the name of the index w_1 from Figure 4.2.

from Figure 4.4, some false proposition at w_3 (i.e. $\overline{Rj}$) is neither false at the situation @, nor does it fail to carry information about John.

To solve this problem, we replace the material conditional 'if ..., then ...' from (4.2.12) and (4.2.17) by the 'starred' conditional '⋆if ..., ⋆then ...' (cf. (4.2.18), (4.2.19), below). The conditional '⋆if ..., ⋆then ...' is interpreted as the definedness relation on partial objects, which orders these objects with respect to their encoded information. In particular, for propositional constants φ and ψ, the conditional '⋆if φ, ⋆then ψ' asserts that the proposition denoted by ψ contains *at least as much information* as φ, such that ψ is true if φ is true, and is false if φ is false. As a result, the conditional from (4.2.18) will be true of w if all true propositions at w_1 which carry information about a are true and if all false propositions at w_1 which carry information about a are false.

(4.2.18) $\big\{\langle w_1, w\rangle \mid \text{for all } p_{\langle s,t\rangle}, {}^\star\text{if } w_1 \in p \text{ and } p \text{ is about } a, {}^\star\text{then } w \in p\big\}$

(4.2.19) $\big\{\langle w_1, w\rangle \mid w \in \varphi \text{ and, for all } p_{\langle s,t\rangle}, {}^\star\text{if } w_1 \in p \text{ and, for some } x, \varphi \text{ is about } x \text{ and } p \text{ is about } x, {}^\star\text{then } w \in p\big\}$

It is easy to see that the intuitive representation of John from Figure 4.4 follows this strategy.

In virtue of the above, the representation of some individual a whose existence is the only available information at the current situation is exactly the type-$\langle s,t\rangle$ representation of a from (4.2.10). This observation is captured in Proposition 4.3:

PROPOSITION 4.3. *If all true propositions at @ which carry information about a are (equivalent to) the proposition 'a exists', then $a^\dagger_{@} = \{w_s \mid a \text{ exists in } w\}$.*

The application of Proposition 4.3 to the single-type semantics of the types $\langle s,t\rangle$ and $\langle s,\langle s,t\rangle\rangle$ will be captured in Chapters 7.3 and 8.3 (Part III).

Notably, the strategy from (4.2.18) also enables the representation of individuals which do not exist in the current situation. This is due to the fact that existence is a bivalent property, such that the existence or non-existence of each individual is decided at every situation. If the non-existence of an individual a is the only true proposition about a at the current situation, we represent a at @ via the set of all situations in which it does not exist.[14] Thus, at the situation w_2 from Figure 4.4, we represent John by the set of situations from (4.2.20):

(4.2.20) $\{w_s \mid w \in \neg Ej\}$

[14]Clearly, to apply this strategy to individuals (e.g. the round square) which do not exist in any situation, we would need to extend the set of situations via *impossible* situations (**Hintikka, 1975**), cf. (**Rantala, 1982**; **Barwise, 1997**; **Zalta, 1997**). However, since the PTQ fragment does not contain designators of the critical individuals, we refrain from adopting this strategy.

The assumption of individual-discerning situations (s.t. there is, for every pair of individuals, a situation at which one, but not the other individual exists; cf. **Ax12**), again preserves the injectivity of this representation.

The specification of information updates (i.e. '$w \in \varphi$') in the strategy from (4.2.19) enables the modeling of information growth. To see this, consider the representation of the proposition 'John runs' at the situation @ from Figure 4.4 (in (4.2.21)). This representation is obtained by *extending* the set of true propositions about John at @ by the information encoded in the proposition 'John runs'. This extension corresponds to the *elimination* of those situations (i.e. @ and w_2) from the set of John-representing situations at @ at which the proposition 'John runs' is either false or undefined:

$$\begin{aligned} (4.2.21) \quad Rj^{\dagger}_{w_1} &= \{w_s \mid \text{John runs at } w\} \cap j^{\dagger}_{@} \\ &= \{w_s \mid \text{John runs at } w\} \cap \{w_s \mid \text{John exists at } w\} \\ &\neq j^{\dagger}_{@} \end{aligned}$$

The above strategy suggests the representation of *properties* of individuals at a given situation by functions from the local representation of individuals in the property's domain (here, the @-specific representation of John from Fig. 4.4) to the local representation of the result of attributing the property to the individuals (e.g. the @-specific representation of 'John runs' from (4.2.21)). The latter representation corresponds to the result of obtaining the representation of John at the better-defined index w_3, which distinguishes itself from @ at most with respect to the obtaining of the proposition 'John runs'.

This completes our identification and description of suitable single basic types. We close by comparing our new single basic types with Partee's preliminary single-type choice, and by summarizing the main results of this chapter.

4.3. Single-Type Candidates and Partee's Conjecture

The introduction to this dissertation (cf. Ch. 1.2.2) has presented Partee's attempt at supporting Proposition 1.2 through the use of extensional properties of Kratzer-style situations (**Kratzer, 1989**), cf. (**Partee, 2006**, p. 40). The considerations from this chapter disclose three interesting facts about this basic-type choice:

(1) The type for *extensional properties of situations* (type $\langle s, t\rangle$) is a suitable single basic type which satisfies Properties (i) to (iv), and which accommodates Propositions 1.2 and 1.3.i.

(2) Partee places more constraints on single-type objects than necessary. Granted her disregard of the behavior of names and sentences from Proposition 1.3, *properties of possible worlds* (i.e. *total* objects of the type $\langle s, t\rangle$) are equally suitable.

(3) Partee neglects an alternative type (for *propositional concepts*, type $\langle s, \langle s, t\rangle\rangle$). This type accommodates Proposition 1.2 *and* its consequences from Proposition 1.3 (including Proposition 1.3.ii).

The observations from items (1) to (3) partly support Partee's basic-type choice. However, they emphasize the need to carefully consider competing candidates (3), and stress the possibility of adhering more closely to Montague's original semantic ontology (cf. the adoption of possible worlds, (2)). The possibility of representing partial situations via sets of their extending possible worlds (cf. Sect. 4.2.4) emphasizes the role of semantic operations and representational strategies in ontology. The failure of an $\langle s, t\rangle$-based single-type semantics to accommodate Proposition 1.3.ii will be the subject of Chapter 7 (Part III).

4.4. Summary

This chapter has identified the types $\langle s, t\rangle$ and $\langle s, \langle s, t\rangle\rangle$ as suitable types for the interpretation of the single basic type o from Part I. These types are associated with propositions and with propositional concepts, respectively. Objects of these two types are familiar from existing work in formal semantics, have an algebraic structure, and enable an injective representation of Montagovian individuals and propositions. In virtue of their specific representational strategy, the types $\langle s, t\rangle$ and $\langle s, \langle s, t\rangle\rangle$ further describe the propositional content of the correlates of individuals in a single-type system.

We have seen that the different representational strategies of the types $\langle s, t\rangle$ and $\langle s, \langle s, t\rangle\rangle$ yield single-type objects of different semantic complexity. Thus, the type $\langle s, \langle s, t\rangle\rangle$ represents individuals as the result of attributing them a contextually salient property (Prop. 4.2, cf. Prop. 1.3.ii). The type $\langle s, t\rangle$ can only represent individuals as the result of attributing them an existence property (Prop. 4.1).

The existence of two differently informative single-type semantics for the PTQ fragment will be the subject of discussion in the next chapter. That chapter will also investigate the distinction between the object- and the metatheory of our single-type semantics, and present several related methodological issues.

CHAPTER 5

Single-Type Methodology

The previous chapter has identified a list of semantically desirable properties for any suitable single basic type. The present chapter describes the suitable types' meta-properties, and discusses a number of related methodological issues. Meta-properties include the equivalence (up to coding) of representations of Montagovian objects in the types $\langle s,t\rangle$ and $\langle s,\langle s,t\rangle\rangle$ with their representations in higher-rank types (Sect. 5.1.1), the existence of two 'brands' of differently informative single-type semantics (Sect. 5.2.1), and the distinction between an object theory and a metatheory of single-type semantics (Sect. 5.3.1). Methodological issues include the robustness of our semantics with respect to their particular single-type choice (Sect. 5.1.2), the natural classification of single-type theories according to their informational strength (Sect. 5.2.2), and their decomposition into differently elementary constituents (Sect. 5.3.2). The distinctions between an object- and a metatheory of single-type semantics and between a weak and a strong representation of Montagovian objects will structure the remainder of this dissertation.

5.1. Robustness in Single-Type Semantics

Our considerations from Chapter 4 have identified the types $\langle s,t\rangle$ and $\langle s,\langle s,t\rangle\rangle$ as the most promising interpreting types for the type o from Part I. This is due to the fact that the types $\langle s,t\rangle$ and $\langle s,\langle s,t\rangle\rangle$ satisfy the requirements from Properties (i) to (vi), such that their associated semantics model the PTQ fragment along the lines of Chapter 3. In particular, the simplicity of the types $\langle s,t\rangle$ and $\langle s,\langle s,t\rangle\rangle$ ensures the (comparatively) low semantic complexity of the interpretations of PTQ expressions in single-type models. However, as has been suggested by our discussion of the types $\langle e,\langle s,t\rangle\rangle$ and $\langle\langle s,t\rangle,t\rangle$ (cf. Ch. 4.2.2), the types $\langle s,t\rangle$ and $\langle s,\langle s,t\rangle\rangle$ are not the only viable single-type choices. Alternatives include the type for functions from situations to generalized quantifiers over situations $\langle s,\langle\langle s,t\rangle,t\rangle\rangle$, from propositions to propositions $\langle\langle s,t\rangle,\langle s,t\rangle\rangle$, and infinitely many others[1].

Section 5.1.1 describes the representations of individuals and propositions in some of these more complex types. Section 5.1.2 discusses the invariance of single-

[1] These types take the form $\langle\langle s,\vec{t}\rangle,\langle\langle s,t\rangle,\vec{t}\rangle\rangle$, where $\vec{t}$ abbreviates constructions of the form $\langle\ldots,\langle\ldots,\langle\ldots,t\rangle,t\rangle,\ldots\rangle$ for any number of types t.

type models under their basic-type choice (granted that the basic types use a same-strength representational strategy and that they satisfy the conditions from Properties (i) to (iii)).

5.1.1. Other Single-Type Choices. Like the types $\langle s,t\rangle$ and $\langle s,\langle s,t\rangle\rangle$, the types $\langle s,\langle\langle s,t\rangle,t\rangle\rangle$ and $\langle\langle s,t\rangle,\langle s,t\rangle\rangle$ enable the principled representation of objects of all Montague types along the lines suggested by Property (iii). In particular, the type $\langle s,\langle\langle s,t\rangle,t\rangle\rangle$ enables the weak representation of individuals and propositions by objects of the form from (5.1.1), resp. (5.1.2), and enables the strong representation of individuals and propositions by objects of the form from (5.1.3), resp. (5.1.4). The objects from (5.1.1) and (5.1.2) code the weak representations from (4.2.10) and (4.2.9). The objects from (5.1.3) and (5.1.4) code the strong representations from (4.2.12) and (4.2.17).

(5.1.1) $\big\{\langle w_s, p_{\langle s,t\rangle}\rangle \mid p = \{w'_s \mid a \text{ exists in } w'\}\big\}$

(5.1.2) $\big\{\langle w_s, p_{\langle s,t\rangle}\rangle \mid p = \{w'_s \mid w' \in \varphi\}\big\}$

(5.1.3) $\big\{\langle w_s, p_{\langle s,t\rangle}\rangle \mid w \in p \text{ and } p \text{ is about } a\big\}$

(5.1.4) $\big\{\langle w_s, p_{\langle s,t\rangle}\rangle \mid (w \in p \text{ or } p = \varphi) \text{ and, for some } x_e,\ \varphi \text{ is about } x \text{ and } p \text{ is about } x\big\}$

The adoption of the basic type $\langle\langle s,t\rangle,\langle s,t\rangle\rangle$ enables an analogous representation of individuals and propositions. In particular, the weak representation of individuals and propositions is described in (5.1.5) and (5.1.6). The strong representation of individuals and propositions is provided in (5.1.7) and (5.1.8). In the last two representations, *approximation* is the inverse of the definedness relation on partial objects from (4.2.18) and (4.2.19). As a result, a proposition φ approximates a proposition ψ iff ψ contains at least as much information as φ.

(5.1.5) $\big\{\langle p_{\langle s,t\rangle}, w_s\rangle \mid a \text{ exists in } w\big\}$

(5.1.6) $\big\{\langle p_{\langle s,t\rangle}, w_s\rangle \mid w \in \varphi\big\}$

(5.1.7) $\big\{\langle p_{\langle s,t\rangle}, w\rangle \mid \text{for all } p'_{\langle s,t\rangle}, \text{ if } p \text{ approximates } p' \text{ and } p' \text{ is about } a, \text{ then } w \in p'\big\}$

(5.1.8) $\big\{\langle p_{\langle s,t\rangle}, w\rangle \mid w \in \varphi \text{ and, for all } p'_{\langle s,t\rangle}, \text{ if } p \text{ approximates } p' \text{ and, for some } x,\ \varphi \text{ is about } x \text{ and } p' \text{ is about } x, \text{ then } w \in p'\big\}$

That the types $\langle s,\langle\langle s,t\rangle,t\rangle\rangle$ and $\langle\langle s,t\rangle,\langle s,t\rangle\rangle$ satisfy the representability requirement from Property (iii) is a direct consequence of the possibility of representing individuals and propositions in the types $\langle s,t\rangle$ and $\langle s,\langle s,t\rangle\rangle$ (cf. Ch. 4.2.3), and of the possibility of coding every object of some Montague type α in the type $\langle\alpha,t\rangle$ (and, hence, in the types $\langle\langle\alpha,t\rangle,t\rangle$, $\langle\langle\langle\alpha,t\rangle,t\rangle,t\rangle$, etc.). The latter is a general rep-

resentational property of type- (or set-)theoretic objects, which is exploited in flexible Montague grammar (cf. Ch. 1.2.1, 1.4.2).

5.1.2. Invariance Properties of Single-Type Models. In virtue of their particular representation strategy, the representations of individuals and propositions in the types $\langle s, \langle\langle s,t\rangle, t\rangle\rangle$ and $\langle\langle s,t\rangle, \langle s,t\rangle\rangle$ will carry the same information as their representations in the type $\langle s,t\rangle$ (resp. $\langle s, \langle s,t\rangle\rangle$), such that they are *equivalent up to coding*. As a result, the provision of a single-type semantics for the PTQ fragment is independent of our particular basic-type choice. (Note, however, that the representation of Montagovian objects in a higher-rank type is typically[2] less intuitive and representationally more 'expensive' than their representation in a lower-rank type. Our adoption of the simplicity requirement from Chapter 4 reflects this consideration).

The independence of single-type semantics of our basic-type choice is reminiscent of the robustness of minimal models[3] of phenomena in mathematics and the exact sciences. This robustness is witnessed by the stability of most mathematical models of physics under calculational shortcuts, and by the formulation-invariance of the minimal logical theories which prove theorems of ordinary mathematics, cf. (**Friedman, 1975**; **Simpson, 2009**).[4]

The irrelevance of our single-type choice for the successful modeling of the PTQ fragment is a particularly strong robustness property. This is suggested by the fact that the existence (up to coding) of a *unique* minimal model of a phenomenon – which is witnessed in single-type semantics – cannot always be assumed in mathematics or in physics. As a result of the strong robustness of our single-type semantics, we need not define a single-type semantics for each of the types $\langle s,t\rangle$, $\langle s, \langle s,t\rangle\rangle$, $\langle s, \langle\langle s,t\rangle, t\rangle\rangle$, $\langle\langle s,t\rangle, \langle s,t\rangle\rangle$, etc. Rather, it suffices to model the PTQ fragment through the use of *one* of these types.

For the purposes of this dissertation, we will identify the paradigmatic basic type with the type $\langle s,t\rangle$ (for the provision of an informationally 'weak' single-type semantics), respectively $\langle s, \langle s,t\rangle\rangle$ (for the provision of a 'strong' single-type semantics). From the semantics of these two types, the single-type semantics of all other types which satisfy Properties (i) to (iii) are then easily obtained.

5.2. Classification of Single-Type Semantics

The previous section has identified the type-invariance of single-type models as an example of a desirable property of minimal models of natural phenomena. Notab-

[2]Exceptions include the 'strong' representation of individuals and propositions in the type $\langle s, \langle\langle s,t\rangle, t\rangle\rangle$. This representation will be discussed in greater detail in Chapter 8.4.

[3]Minimal models are models which account *exactly* for the observed phenomena (and for no other phenomena).

[4]I owe this observation to Sam Sanders.

ly, the former is not the only desirable property of the presented models. Other properties include the invariance of single-type models under the replacement of Montague types by their equivalent types from other type systems (e.g. the replacement of $\langle s,t\rangle$ by the type for semantically primitive propositions p (**Thomason, 1980**), cf. (**Chierchia and Turner, 1988**; **Fox et al., 2002**; **Pollard, 2005**)), and the natural classification of representations of Montagovian objects according to their informational strength. In particular, this classification is witnessed by the bifurcation of single-type semantics into *weak* theories of linguistic meaning (which interpret proper names as the result of attributing their type-e referent an existence property) and *strong* theories of meaning (which interpret names as the result of attributing their referent a (set of) contextually salient property(-ies)).

Below, we will first discuss the distinction between the 'weak' and 'strong' representations of Montagovian objects from Chapter 4 (in Sect. 5.2.1). We will then show that the resulting classification of single-type theories is a particular instance of a general classification of minimal theories in other sciences (Sect. 5.2.2).

5.2.1. Weak and Strong Single-Type Semantics. Our discussion of the representation of individuals and propositions in the type $\langle s,t\rangle$ has already suggested the possibility of distinguishing between two differently 'rich' representation strategies. Thus, we can either represent an individual a by its *minimal globally* associated information, or by its *maximal locally* (i.e. contextually) associated information. The former strategy corresponds to the identification of (some coding of) the proposition 'a exists'. The latter strategy corresponds to the identification of the intersection of all true propositions at some situation which carry information about a. To ensure its situation-generality, the rich representation of individuals proceeds in the type $\langle s,\langle s,t\rangle\rangle$.

The different representations of individuals are given in the left column from Figure 5.1. They pair with the representations of propositions in the right column to yield informationally *weak*, resp. *strong* interpretations of names and sentences.

	individuals a (type-e)	**propositions** φ (type-$\langle s,t\rangle$)
weak (type-$\langle s,t\rangle$)	$\{w_s \mid a \text{ exists in } w\}$	$\{w_s \mid w \in \varphi\}$
strong (type-$\langle s,\langle s,t\rangle\rangle$)	$\{\langle w_1,w\rangle \mid$ for all $p_{\langle s,t\rangle}$, *if $w_1 \in p$ & p 'is about' a, *then $w \in p\}$	$\{\langle w_1,w\rangle \mid w \in \varphi$ & $\forall p$, *if $w_1 \in p$ & for some x, φ 'is about' a & p 'is about' x, *then $w \in p\}$

FIGURE 5.1. 'Weak' and 'strong' representations of individuals/propositions.

The representations of basic Montagovian objects from Figure 5.1 are the only *natural* candidates: Our characterization of property attribution as controlled information growth (cf. (4.2.21)) excludes 'hybrid' representation pairs whose members exhibit different levels of informational complexity. In particular, the exclusion of the *weak–strong* pair (4.2.10)–(4.2.19) (or of the pair consisting of $\{\langle w_1, w\rangle \mid a \text{ exists in } w\}$ and (4.2.19)) is justified by the fact that the attribution of some property F to the individual a will *only* extend the type-$\langle s,t\rangle$ representation of a by the information encoded in the proposition Fa (and by the information of no other non-trivial proposition). As a result, the proposition Fa is represented by the characteristic function of the set of situations $\{w_s \mid w \in Fa\}$.

Our argument for the exclusion of the *strong–weak* pair (4.2.18)–(4.2.9) (or of the pair (4.2.18)–(4.2.13)) follows the inverse strategy: Since the attribution of the property F to a *extends* the type-$\langle s,\langle s,t\rangle\rangle$ representation of a at the situation @ by the information of Fa,[5] it cannot hold that $(Fa_@)^\dagger \supseteq a^\dagger_@$.

As a result, all single-type representations of individuals and propositions will either follow the *weak* representation strategy from the top row in Figure 5.1, or the *strong* representation strategy from the bottom row in Figure 5.1. Their associated objects will encode weak (i.e. 'poor', or simple), respectively strong (i.e. 'rich', or complex) information about the represented Montagovian objects. The single-type semantics from Part III (cf. Ch. 7, 8) inherit their names from the informational strength of their associated objects.

5.2.2. Classification Properties of Single-Type Models. The classification of single-type theories according to their objects' informational strength follows the classification of logical theories according to their proof-theoretic strength. In the area of Reverse Mathematics, cf. (**Friedman, 1975**), researchers have observed that the minimal axiom systems which prove theorems of ordinary mathematics (called the 'Big Five' in (**Simpson, 2009**)) are totally ordered with respect to their proof-theoretic strength.

Theories of single-type semantics exhibit a similar ordering as the 'Big Five'. Only, rather than being ordered with respect to their ability to prove their equivalence with a particular mathematical theorem, these theories are ordered with respect to their ability to identify a name's equivalent sentences. Thus, while the proper name Barbara Partee will only be equivalent to the sentence Barbara Partee exists in a *weak* single-type semantics (cf. Ch. 7.3), it may, depending on the relevant situation, be equivalent to the sentence Barbara Partee is entering the room (or Barbara Partee is arriving) in a *strong* single-type semantics.

This ends our identification of different subclasses of single-type semantics.

[5]This is conditional on the compatibility of F with a's other properties at w (s.t. there is no property P in the set of a's properties at w of which it holds that $\big(\{x_e \mid x \in P\} \cap \{y_e \mid y \in F\}\big) = \emptyset$).

We next turn to a distinction between the semantics' different theoretical levels.

5.3. Levels of Single-Type Semantics

Our previous considerations have identified the types $\langle s,t\rangle$ and $\langle s,\langle s,t\rangle\rangle$ as the paradigmatic single basic types for the modeling of the PTQ fragment along the lines of (**Partee, 2006**). To show the representability of the types $\langle s,t\rangle$ and $\langle s,\langle s,t\rangle\rangle$, Chapter 4 has given algorithms for the representation of individuals and propositions in objects of each of these two types (cf. Sect. 4.2.3). Thus, in the type $\langle s,t\rangle$, individuals and propositions are represented by the set of situations in which the individual exists, resp. at which the proposition's designator is true. In the type $\langle s,\langle s,t\rangle\rangle$, individuals are represented by functions from situations w to the set of situations at which the designators of all true propositions at w which carry information about the individual are true. Propositions φ are represented by functions from w to the intersection of the set of situations at which φ is true and the type-$\langle s,\langle s,t\rangle\rangle$ representations of φ's aboutness subjects at w.

Clearly, the resulting representations constitute objects of the type $\langle s,t\rangle$, or $\langle s,\langle s,t\rangle\rangle$. We will see in Part III that the semantics of these two types interpret all logical PTQ forms into constructions out of the types $\langle s,t\rangle$, respectively $\langle s,\langle s,t\rangle\rangle$. Notably, however, the descriptions of single-type objects still contain expressions of lower-rank types, which are formed from Gallin's basic types e, s, and t. For example, the description of the weak representation of the individual a from (4.2.10) relies on the availability of individuals (there, the individual a) and situations (w). The description of the strong representation of a from (4.2.18) further relies on the availability of propositions (p).

Below, we will first identify the rationale behind the use of lower-rank types in the metatheory of our single-type semantics (in Sect. 5.3.1). We will then show that this rationale warrants a distinction between the object theory and the metatheory of any sufficiently expressive single-type semantics (in Sect. 5.3.2).

5.3.1. Constraining Single-Type Representations. The multi-typed description of single-type objects is required by the *underdefinedness* of the values of the translations of PTQ words from Definition 3.2.2. As a result of this underdefinedness, translations of logical PTQ forms may receive an interpretation into TY_0 objects which disqualify as "reasonable candidate[s] for interpretations of English", cf. (**Montague, 1973**, p. 263). For example, if we failed to constrain the interpretation of the TY_0 term ***john*** by the requirement $[\![\boldsymbol{john}]\!] = \{w_s \mid \text{John exists in } w\}$, nothing would prevent the association of John with the intuitive representation, $\{w_s \mid \text{Mary exists in } w\}$, of Mary. But this is arguably undesirable.

By specifying, for every single-type term, which element in the relevant Gallin-style model it designates, descriptions of the form from (4.2.10) and (4.2.18)

take the role of semantic constraints. These constraints differ from Montagovian meaning postulates (**Montague, 1973**), cf. (**Carnap, 1988**, Supp. B), with respect to their formulation in the *meta*language, and to their comparatively greater expressive power: In contrast to meaning postulates, constraints are not concerned with the specification of linguistically significant aspects of lexical meaning. Rather, they provide an algorithm for the 'right' model-theoretic interpretation of a given TY_0 term. For the translations of PTQ words in the types $\langle s,t\rangle$ and $\langle s,\langle s,t\rangle\rangle$, such constraints will be given in Definitions 7.2.2 and 8.2.2 (Part III).

The need to describe the representations of Montagovian objects through expressions of a lower type imposes an important restriction on Partee's conjecture:

PROPOSITION 5.1 (Levelling constraint). *The 'correct' interpretation of logical PTQ forms in single-type semantics requires a multi-typed metatheory.*

The use of a multi-typed metatheory does not compromise the integrity of our single-type semantics: The semantics from Chapters 7 and 8 enable the interpretation of *all* logical PTQ forms into '*pure*' single-type objects. Further, since the single-type interpretations of existential sentences (e.g. Bill exists) adopt the representation strategy for individuals in a weak single-type semantics (here, the strategy for the representation of Bill; cf. Prop. 4.1, 4.3), the distinction between the lower-type description of the single-type interpretations of proper names and sentences does not coincide with the Montagovian distinction between individuals and propositions.

We will see in Part III that the restriction of single-type interpretations through multi-typed semantic constraints enables the identification of their designators' semantic equivalents.

5.3.2. Stratification Properties of Single-Type Models. To capture the difference between the types of our single-type objects (cf. Part I) and the types of the terms in their defining semantic constraints (cf. Ch. 4.2.3), we will hereafter distinguish the *object theory* of our single-type semantics (at which we find the types of single-type PTQ translations; cf. Def. 3.2.2, Def. 7.2.1) from the *metatheory* of our single-type semantics (at which we give the translations' definitions; cf. Def. 7.2.2, 8.2.2). While the type system of the object theory will only contain constructions out of the type $\langle s,t\rangle$, respectively $\langle s,\langle s,t\rangle\rangle$, the type system of the metatheory will contain (equivalents of) all Montague types.

The distinction between an object- and a metatheory of our single-type semantics is orthogonal to the distinction between a weak and a strong representation of Montagovian objects. As a result, our single-type semantics from Part III will provide a separate discussion of the object theories of our weak and strong single-type semantics. The same applies to our discussion of the two systems' application to Montague's PTQ fragment: Since the semantics of the types $\langle s,t\rangle$ and

$\langle s, \langle s, t\rangle\rangle$ will define their objects through the use of expressions from Gallin's logic TY_2, the metatheory of our *weak* single-type semantics (cf. Ch. 6) will be identified with the metatheory of our *strong* semantics.

We close this section by identifying the effect of our use of layered structures on the strategy of indirect interpretation.

5.3.3. Iterated Indirect Interpretation. The introduction to this dissertation (Ch. 1.1.1; cf. Part I) has announced an *indirect* interpretation of the PTQ fragment, which proceeds via an intermediate one-typed language. Thus, to provide the 'pure' single-type semantics from Part I, we have first translated all logical PTQ forms into TY_0 terms (cf. step 3). The semantic values of these forms have then been obtained via the model-theoretic interpretations (cf. step 2.i) of the forms' TY_0 translations. For easy reference, a variant of the diagram of indirect representation from Figure 1.1 is reprinted in Figure 5.2:

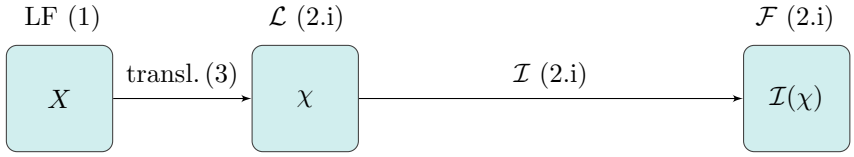

FIGURE 5.2. Indirect interpretation in the 'pure' single-type model.

The need to constrain the single-type interpretations of PTQ forms (cf. Sect. 5.3.1) requires the introduction of a further step in this interpretation process: Instead of interpreting the single-type translations of logical PTQ forms *directly* into their associated objects (cf. step 2.i), we first need to define the translations into terms of the metalogic TY_2 (step 4). The semantic values of PTQ forms will then be obtained *indirectly* via the interpretations of their translations' *definitions* (step 2.ii). The adopted interpretation procedure is thus *doubly indirect* (or (*once*) *iterated*).

The doubly indirect interpretation of PTQ forms is captured in Figure 5.3 (next page). In the figure, the new steps (2.ii) and (4) are framed. The term χ° is the metalogical definition of the single-type term χ. The designated language, frame, and interpretation function of the logic TY_2 are denoted by $\mathcal{L}^2$, $\mathcal{F}^2$, and $\mathcal{I}^2$.

Admittedly, the iterated translation of logical PTQ forms (first into single-type terms, and then into their TY_2 definition) is rather cumbersome. To simplify this process, one could instead translate PTQ forms *directly* into suitably typed TY_2 terms.[6] The resulting interpretation process enables the provision of a single-type semantics *without a single-type logic*. This process is captured in Figure 5.4.

[6] I owe this observation to Jeroen Groenendijk.

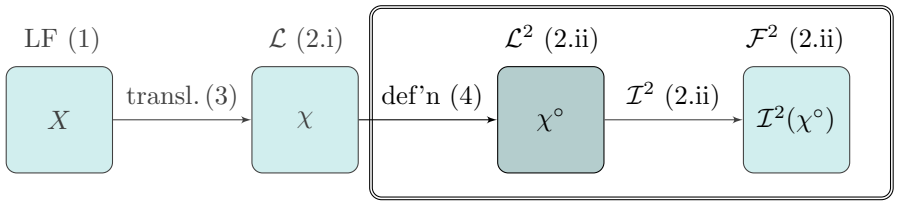

FIGURE 5.3. Defined doubly indirect interpretation.

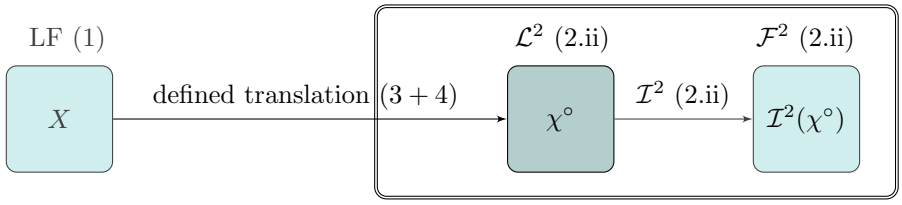

FIGURE 5.4. Defined singly indirect interpretation.

The formulation of single-type semantics in the familiar logic TY_2 complies with the principle of parsimony from the introduction to this dissertation (here interpreted as methodological parsimony). However, the remainder of this dissertation will adopt the method of doubly indirect interpretation from Figure 5.3. This choice is supported by the possibility of justifying the definitions of the defined single-type connectives from Notation 2.1.1 (1), by the greater ease of recognizing the single-type correspondents of logical PTQ forms (2), and by the use of iterated translation in other areas of formal syntax and semantics (3). We discuss these reasons in their order of mention:

(1) In Chapter 2.1 (cf. Nota. 2.1.1), we have defined single-type stand-ins of the familiar truth-functional connectives and quantifiers from the TY_0 proxies, $\textcircled{\perp}$ resp. $\dot{\Rightarrow}$, for falsum and logical implication. We have motivated these definitions with reference to their analogy with Henkin's (**1950**) definitions of the truth-functional connectives and quantifiers from $\perp$ and $\Rightarrow$. However, since we have been unable to formulate easy truth- or falsity-conditions for basic single-type terms, it has been impossible to check these definitions.

The 'translation' of single-type constants into their TY_2 definitions (along the lines of Fig. 5.3; steps 4 and 2.ii) enables such a test. In particular, we can use the 'translations' (or definitions) of the single-type constants $\textcircled{\perp}$ and $\dot{\Rightarrow}$ and the conventions from Notation 2.1.1 to obtain the TY_2 definitions of the remaining single-type connectives: If the thus-obtained definitions match the postulated translati-

ons, the definitions of the single-type connectives from Notation 2.1.1 are correct. If the definitions differ from the translations, the definitions of the single-type connectives from Notation 2.1.1 are flawed, and must be rejected.

For example, since our 'weak' single-type logic from Chapter 7 defines the terms $\textcircled{\scriptsize\bot}$, $\textcircled{\scriptsize\top}$, and $\dot{\Rightarrow}$ as the designators of (the characteristic functions of) the sets $\emptyset$, $\bigcup_{w\in W} w$, and $\{\langle \boldsymbol{x}_{\langle s,t\rangle}, \boldsymbol{y}_{\langle s,t\rangle}, w_s\rangle \mid \text{if}^{\dot{\forall}}\ w \in \boldsymbol{x},\ \text{then}^{\dot{\forall}}\ w \in \boldsymbol{y}\}$ (where 'if$^{\dot{\forall}}$ ..., then$^{\dot{\forall}}$ ...' denotes the relation of logical implication), and since we have defined $\textcircled{\scriptsize\top}$ as $(\textcircled{\scriptsize\bot} \dot{\Rightarrow} \textcircled{\scriptsize\bot})$ (s.t. the replacement of $\textcircled{\scriptsize\bot}$ and $\dot{\Rightarrow}$ in $(\textcircled{\scriptsize\bot} \dot{\Rightarrow} \textcircled{\scriptsize\bot})$ by their definitions yields the designator of the set $\bigcup_{w\in W} w$, which is the intuitive translation of $\textcircled{\scriptsize\top}$), the definition of $\textcircled{\scriptsize\top}$ is correct. Other examples of this test will be given in Chapter 7.2.2 (cf. (7.2.1)) and in Appendix C.3.

(2) Beyond verificational advantages, the inclusion of a level of a designated single-type language in our interpretation procedure facilitates the identification of the single-type correspondents of logical PTQ forms. In Chapter 4, we have seen that the single-type interpretations of proper names (e.g. Bill) will be associated with more complex objects than their traditional Montagovian counterparts (e.g. with the characteristic function of the set $\{w_s \mid \text{Bill exists in } w\}$). We expect that expressions from other syntactic categories will receive an interpretation into even more complex objects. For example, in the $\langle s,t\rangle$-based semantics from Chapter 7, the verb walks will be associated with the characteristic function of the set from (5.3.1). This function sends the type-$\langle s,t\rangle$ representation of every individual a to the proposition 'a walks':

$$\{\langle \{w_1 \mid a \text{ exists in } w_1\}, \{w_s \mid a \text{ walks at } w\}\rangle \mid a \in D_e\} \tag{5.3.1}$$

The assumption of a primitive correspondent for the above objects aids our conception of these objects as a single unit of interpretation. Their designators in a single-type language (e.g. in the language $\mathcal{L}$ from Part I) can then serve as 'abbreviations' for the descriptions of these objects in the language of the metatheory TY_2. Thus, the designation of the object from (5.3.1) via the single-type term ***walk*** is significantly shorter than the object's describing TY_2 term (cf. Def. 7.2.2, MP5). The use of a designated single-type language further supports the conception of our single-type semantics as an *intended* theory of meaning for the PTQ fragment, rather than as a *peculiar coding* of its Montagovian semantics.

(3) The use of an iterated translation (or interpretation) procedure is not uncommon in formal syntax and semantics. For example, to enable a machine translation between different natural languages, Rosetta[7] (**1994**) invoke a number of intermediate languages, including controlled versions of Dutch and Spanish, cf.

[7]The name 'Rosetta' stands for a collective of twelve authors: Jan Landsbergen, Lisette Appelo, Franciska de Jong, Theo Janssen, Jan Odijk, André Schenk, Joep Rous, René Leermakers, Harm Smit, Petra de Wit, Elly van Munster, and Elena Pinillos Bartolomé.

(**Partee, 1997**, p. 24). To interpret a fragment of natural language in models of Property Theory with Curry Typing (PTCT), Fox and Lappin (**2004**) similarly invoke first the object language and then the (extended) metalanguage of their logic PTCT.

This completes our motivation for using the method of doubly indirect interpretation in a singe-type semantics. We close the present chapter with a summary of its main results.

5.4. Summary

This chapter has discussed three salient methodological features of single-type semantics with the candidate types from Chapter 4: their robustness with respect to the particular basic-type choice, their classification according to their objects' informational strength, and their stratification into different theoretical levels. In virtue of the first two features, representations of Montagovian objects in the types $\langle s, t\rangle$ or $\langle s, \langle s, t\rangle\rangle$ are informationally equivalent (up to coding) to their representations in higher types, and are associated with two differently informative theories of single-type semantics. In virtue of the last feature, we distinguish the metatheory from the object theory of our single-type semantics.

The above features identify our semantics' merits and limitations. In particular, the robustness and (self-)classification of our single-type theories – which are both desirable methodological properties – suggest the 'naturalness' of the modeled phenomenon. The impossibility of constraining the interpretations of logical PTQ forms in a 'pure' single-type semantics identifies an important limitation on any single-type semantics: Unless the semantics employs multi-typed semantic constraints (and is, thus, 'impure'), it will not be able to accommodate the observations from Proposition 1.3.

The theories of meaning from Part III are 'impure' (or '*mixed*') single-type semantics of the above sort. Chapter 7 presents a weak $\langle s, t\rangle$-based single-type semantics. A strong single-type semantics based on the type $\langle s, \langle s, t\rangle\rangle$ will be developed in Chapter 8.

Part III

'Mixed' Single-Type Semantics

This part of the dissertation presents a *weak* and a *strong* single-type semantics for Montague's PTQ fragment in the sense of Chapter 5.2. These semantics are *mixed* (or '*impure*') theories of linguistic meaning, which analyze the TY_0 translations of logical PTQ forms as constructions out of propositions, respectively of propositional concepts. Proper names, sentences, and complement phrases will all receive an interpretation in these types. In addition to the merits of the 'pure' single-type semantics from Part I, the proposed semantics provide easy truth-conditions for proper names (cf. Prop. 1.3.i) and identify the semantic content of names with the content of certain sentences.

To provide the fragment of English from (**Montague, 1973**) with a 'mixed' single-type semantics, we use the method of doubly indirect interpretation from Chapter 5.3.3 (cf. Fig. 5.3). Thus, to obtain an $\langle s,t\rangle$- or $\langle s,\langle s,t\rangle\rangle$-based semantics for the fragment, we first adapt all relevant concepts of the logic TY_0 to constructions out of the types $\langle s,t\rangle$ and $\langle s,\langle s,t\rangle\rangle$ (cf. step 2.i). We then specify a set of semantic constraints whose members ensure the interpretation of proper names and sentences into objects of the form from Chapter 4.2.3 (cf. step 4). We will describe a logic with the single basic type $\langle s,t\rangle$ in Chapter 7.1. A logic with the basic type $\langle s,\langle s,t\rangle\rangle$ will be the subject of Chapter 8.1. The later sections of Chapters 7 and 8 translate logical PTQ forms into terms of each of these two logics (cf. step 3).

We have noted in Chapter 5.3 that the possibility of specifying the particular single-type values of linguistic expressions relies on the availability of a multi-typed metatheory (cf. Prop. 5.1). In reflection of this consideration, we precede our development of the particular single-type logics with a presentation of the logic TY_2^3 (cf. step 2.ii; in Ch. 6). This logic is a partial streamlined variant of Montague's logic IL.

CHAPTER 6

The Metatheory, TY_2^3

The present chapter defines the metatheory of our single-type semantics in the form of the logic TY_2^3. This logic is a three-valued three-sorted type logic, whose models contain partial objects of the form described in Chapter 4.2.4.

The name of the logic, 'TY_2^3', follows Gallin's (**1975**) convention of subscripting a logic's name by the number of its basic types, not counting the truth-value type t.[1] For every natural number n, we thus let 'TY_n' denote a logic with $n+1$ basic types (granted the existence of the basic type t). Correspondingly, the logic TY_0 (with the basic type t) is Henkin's Theory of Propositional Types, cf. (**Henkin, 1963**)[2], the logic TY_1 (with the basic types e and t) is Church's Simple Theory of Types, cf. (**Church, 1940**), and the logic TY_2 (with the basic types e, s, and t) is Gallin's (**1975**) streamlined variant of Montague's Intensional Logic, cf. (**Montague, 1973**). For the present purposes, we adopt a partial n-ary variant of the logic TY_2 that is inspired by (**Muskens, 1995**, Ch. 6). The partiality of TY_2^3 models is indicated by the superscript of the logic's name.

The chapter is organized as follows: Section 6.1 defines the types and terms of the logic TY_2^3. Terms of this logic include an existence predicate, the iota operator, and other special-purpose operators which enable the definition of the designated single-type constants from Part I. Section 6.2 introduces general TY_2^3 models, and identifies the subclass of De Morgan models. Sections 6.3 and 6.4 provide a Tarski-style truth-definition for TY_2^3 terms, and define the relations of TY_2^3 equivalence and entailment.

6.1. Types and Terms

The basic types of the logic TY_2^3 are Gallin's types for *individuals* e, *indices* s, and *truth-values* t. Since we will provide a partial interpretation of the types s and t (cf. Ch. 4.2.4), we will hereafter refer to objects of these two types as '*situations*' and '*truth-combinations*', respectively. From e, s, and t, the types of the logic TY_2^3 are obtained via a generalized variant of the rule **CT** as follows:

[1] The exclusion of the type t from the 'counted' types is justified by the fact that quantification and binding typically do not apply to type-t objects.

[2] In Part I, we have used 'TY_0' instead as the name of the simplest single-type logic.

DEFINITION 6.1.1 (TY_2^3 types). The set $\mathsf{2Type}$ of TY_2^3 *types* is the smallest set of strings such that, for $0 \leq n \in \mathbb{N}$, if $\alpha_1, \ldots, \alpha_n \in \mathsf{2Type}$ and $\alpha_{n+1} \in \{e, s, t\}$, then $(\alpha_1 \ldots \alpha_n; \alpha_{n+1}) \in \mathsf{2Type}$.

In analogy to Definition 2.1.1, Definition 6.1.1 describes complex types as the types for functions from n-tuples of objects of the types $\alpha_1, \ldots, \alpha_n$ to objects of a basic type. The definition of TY_2^3 types as constructions to one of the *basic* TY_2^3 types e, s, or t guarantees a correspondence between unary and multi-argument functions, cf. (**Schönfinkel, 1924**).

To ensure an algebraic structure on TY_2^3 domains, we will hereafter focus on a proper subclass of TY_2^3 types whose members are called *conjoinable types*. The set of conjoinable TY_2^3 types is defined as follows, cf. (**Partee and Rooth, 1983**, p. 4):

DEFINITION 6.1.2 (Conjoinable TY_2^3 types). The set CoType of *conjoinable types* of the logic TY_2^3 is the smallest set of strings such that, for $0 \leq n \in \mathbb{N}$, if $\alpha_1, \ldots, \alpha_n \in \mathsf{2Type}$, then $(\alpha_1 \ldots \alpha_n; t) \in \mathsf{CoType}$.

According to the above, a TY_2^3 term has a conjoinable type if its type is either the type for truth-combinations t or a construction to the type t (via the rule from Def. 6.1.1). In these two cases, we say that the term *is conjoinable*.

The restriction to conjoinable TY_2^3 types implements the algebraicity requirement from Chapter 4.1.1 and enables the partialization of suitable single-type candidates from Chapter 4.2.4. We will see below that the conjoinability of single-type types further obviates the need for single-type stand-ins of $\wedge, \vee$, and $\neg$.

We next define a class of languages for the logic TY_2^3.

A language L^2 for TY_2^3 is a countable set $\cup_{\alpha \in \mathsf{2Type}} L_\alpha$ of uniquely typed non-logical constants, which is supplemented by a countable set $\mathcal{V}^2 := \cup_{\alpha \in \mathsf{2Type}} \mathcal{V}_\alpha$ of uniquely typed variables. We assume that L^2 includes constants for the false formula $\bot$ (*falsum*) and the neither-true-nor-false (or 'undefined') formula $*$ (called '*star*'). From these expressions, complex TY_2^3 terms are formed inductively with the help of application and abstraction, the constant for logical implication $\Rightarrow$, and the non-logical constants E, ι, and *abt*.

DEFINITION 6.1.3 (TY_2^3 terms). Let $\alpha_1, \ldots, \alpha_n, \beta \in \mathsf{2Type}$, and let $\alpha_{n+1} \in \{e, s, t\}$. Let $\epsilon \in \mathsf{CoType}$. The set T_α^2 of TY_2^3 *terms* of type α is then defined thus:

(i) $L_\alpha^2, \mathcal{V}_\alpha^2 \subseteq T_\alpha^2$, $*, \bot \in T_t^2$;
(ii) If $B \in T^2_{(\beta\alpha_1 \ldots \alpha_n; \alpha_{n+1})}$ and $A \in T_\beta^2$, then $(B(A)) \in T^2_{(\alpha_1 \ldots \alpha_n; \alpha_{n+1})}$;
(iii) If $A \in T^2_{(\alpha_1 \ldots \alpha_n; \alpha_{n+1})}$ and $x \in \mathcal{V}_\beta^2$, then $(\lambda x.A) \in T^2_{(\beta\alpha_1 \ldots \alpha_n; \alpha_{n+1})}$;
(iv) If $B, C \in T_\epsilon^2$, then $(B \Rightarrow C) \in T_t^2$;
(v) If $A \in T_e^2$ and $w \in T_s^2$, then $E(A, w) \in T_t^2$;
(vi) If $\phi \in T_t^2$ and $x \in \mathcal{V}_\alpha^2$, then $(\iota x.\phi) \in T_\alpha^2$;

(vii) If $A \in T_e^2$ and $\varphi \in A \in T_{(s;t)}^2$, then $abt(A, \varphi) \in T_{(s;t)}^2$.

Clause (i) identifies all suitably typed non-logical constants and variables as terms of the logic TY_2^3. Clauses (ii) and (iii) assert the T_α^2-membership of the results of functional application and lambda abstraction (cf. Ch. 2.1). Clauses (iv) to (vii) specify the formation of complex TY_2^3 terms.

The *existence predicate* E from clause (v) enables our definition of weak single-type objects (in Ch. 7; cf. Ch. 4.2.3) and our definition of the single-type translation of the verb exists (cf. Ch. 7, 8). This predicate is a non-logical constant of the type $(e\, s; t)$ which asserts the existence of a given individual at the specified situation.[3] Thus, the term $E(A, w)$ (read 'A *exists in* w') asserts that the situation w is inhabited by the individual denoted by A. The introduction of a designated predicate E is required by the fact that the TY_2^3 quantifiers $\exists$ and $\forall$ range over a single situation-independent domain of *possible*[4] individuals, which exist in some situation. As a result, the TY_2^3 formula $\exists x.x = A$ (read '*there is an A [at some situation]*') does not imply the existence of the individual denoted by A in the *current* situation. Only the predicate E enables us to refer to the set of inhabitants of a given situation (i.e. to the *actual* individuals in that situation).

The behavior of E is governed by the axioms **Ax10** to **Ax12**. Below, the variables x and y range over individuals. The variables i and j range over situations. The connectives $\top, \forall, \exists, =, \neg, \neq, \wedge$, and $\vee$ have their usual definitions:

Ax10 (Bivalence)**.** $\forall x \forall i.E(x, i) \vee \neg E(x, i)$

Ax11 (Trans-situation identity)**.** $\exists x \exists i \exists j.i \neq j \wedge (E(x, i) \wedge E(x, j))$

Ax12 (Ontological independence)**.** $\forall x \forall y.x \neq y \Rightarrow (\lambda i.E(x, i)) \neq (\lambda j.E(y, j))$
i.e. $\forall x \forall y.x \neq y \Rightarrow \big(\exists i.(E(x, i) \wedge \neg E(y, i)) \vee (E(y, i) \wedge \neg E(x, i))\big)$

Axiom **Ax10**, cf. (**Muskens, 1995**, p. 71), defines the existence predicate E as a bivalent predicate, which is either true or false of every individual/situation pair.[5] Axiom **Ax11** asserts that some individuals exist in *more than one* situation, such that these individuals are *identical* across situations. In particular, the trans-situation identity of individuals obviates the introduction of their Lewisian counterparts, cf. (**Lewis, 1986**). Axiom **Ax12** asserts that individuals are discernible by

---

[3] In quantified modal logic, cf. (**Linsky and Zalta, 1994**; **Zalta, 1983**), this predicate (written '$C!$') asserts the concreteness of the relevant individual at the situation.

[4] This motivates the description of $\exists$ and $\forall$ as *possibilist* quantifiers, cf. (**Scott, 1979**).

[5] One can omit **Ax10** by *defining* the term $\lambda x \lambda i.E(x, i)$ as $\lambda x \lambda i \exists b.x = b(i)$, where b is a variable over individual concepts (type $(s; e)$) which does not contain any free occurrences of x. The presented definition of E generalizes the familiar definition of the existence predicate in free and intuitionistic logic, cf. (**Lambert, 1960**; **Scott, 1979**). However, since we assume that the behavior of E is further constrained by **Ax11** and **Ax12**, we introduce E instead as a primitive.

the situations in which they exist: For every pair of individuals, there is a situation at which one, but not the other individual exists.

In Chapter 4.2.3, we have motivated our adoption of axiom **Ax12** with reference to the need to ensure an injective relation between individuals and their type-$(s;t)$ representations. However, this axiom is more than a technical trick to satisfy the representability requirement from Chapter 4.1.2: It formulates (a strengthened version of) the assumption of contingent existence for individuals from modal logic. This assumption is made in (**Kripke, 1963**, p. 65):

> [...] of course, $\varphi(\mathbf{H})$ [the domain of the possible world $\mathbf{H}$] need not be the same set for different arguments $\mathbf{H}$, just as, intuitively, in worlds other than the real one, some actually existing individuals may be absent, while new individuals, like Pegasus, may appear.

The discernibility of individuals by the different situations in which they exist is analogous to the discernibility of propositions by the different situations at which their designating formulas are true (resp. false): Just like two non-equivalent formulas will never be true (or false) *at* exactly the same situations, two non-identical individuals will never exist *in* exactly the same situations.

The ontological independence of individuals is compatible with Kripke's 'necessity of origin' thesis, cf. (**Kripke, 1980**, p. 113).[6] According to this thesis, all living organisms have their biological origin by necessity. In particular, since all zygotes originate from the joining of two gametes, the existence of every animal presupposes the existence of its parents (s.t. $\neg\exists x.(\exists i.E(x,i)) \wedge (\exists y.\mathit{parent}(x,y) \wedge (\neg\exists j.E(y,j)))$). The 'necessity of origin' thesis thus blocks one of the disjuncts of the second formulation of **Ax12**. However, since the existence of potential parents is not conditional on their production of offspring (s.t. the ontological dependence is only one-directional), the necessity of origin does not violate axiom **Ax12**.

Axiom **Ax12** is further supported by the discernibility of non-identical individuals[7] (in (6.1.1), below) and by the possibility of forming the individual domain of 'new' situations by identifying the individuals which witness a given property at another situation.

$$\forall x \forall y.x \neq y \Rightarrow \left(\exists P_{(e\,s;t)} \exists i.P(x,i) \wedge (\neg P(y,i) \vee P(y,i) = *)\right) \tag{6.1.1}$$

The combination of these two principles provides a mechanism for the construction of individual-discerning situations: In particular, by constructing a situation which is inhabited by exactly all individuals which witness the property that discerns a particular individual from another individual at the relevant situation i,

[6] I owe this observation to Ed Zalta.

[7] This principle – which is weaker than **Ax12** – extends to classical abstract individuals in the sense of (**Zalta, 1999a**), cf. (**Zalta, 1999b**, §4.50).

we obtain a situation which E-discerns these two individuals. For example, since, in the situation w_1 from Figure 4.2, the property 'running' is only true of John (and is false of Mary), we can construct a new situation in which only the runners at w_1 (and, thus, John, but not Mary) exist. This situation can take the form of a situation at which only the designator of the proposition 'John exists' is true, or at which the designators of the propositions 'John exists' and 'John runs' (or of the propositions 'John exists' and 'John does not run') are true, etc. Since Mary does not exist in this situation, this situation E-discerns John from Mary.

Granted the above principles, the *in*ability to construct an E-discerning situation for two non-identical individuals generates a contradiction.

Admittedly, axiom **Ax12** prevents the accommodation of individuals (like Carroll's (**2000**) twins Tweedledum and Tweedledee, or entangled photons) whose existence in each situation is conditional on the existence of the other element in the pair (s.t. the ontological dependence between these individuals is *bi*-directional).[8] Individuals of this kind preempt the strategy for the construction of 'discerning' situations from the second-to-last paragraph. But the unavailability of E-indiscernible individuals is not a serious impediment to the metatheory of our single-type semantics. This is due to the fact that entanglement is a very unstable state, and that the possibilist existence of E-indiscernible individuals (e.g. Tweedledum and Tweedledee) is not a natural assumption in the ontology of linguistic semantics. In particular, every formula which *introduces* two distinct E-indiscernible individuals has a higher or equal complexity to Π^1_1.[9] But the assumption of such a formula far exceeds the strength of ordinary linguistic statements.

Beyond the above, the adoption of axiom **Ax12** is justified by our characterization of situations as *partial* possible worlds (cf. Ch. 4.2.4). Following (**Kratzer, 2011**, Sect. 7, 9), we can thus assume the existence of *minimal situations* w at which only (the entailments of) a single atomic sentence (e.g. Tweedledum dances) are true. Since the truth of such a sentence at w only presupposes the existence of the sentence's NP subject (e.g. the existence of Tweedle*dum*) and its satisfaction of the property denoted by the VP, the situation w discerns the individual denoted by the NP subject from all other individuals.

This completes our motivation for the behavior of the existence predicate E. The iota operator from Definition 6.1.3, clause (vi) further enables the definition of weak single-type objects (in Ch. 7). In particular, TY^3_2 terms of the form $\iota x_\alpha.\phi$ where ϕ is an open formula in which only x is free (read '*the unique x s.t.* ϕ') identify the designator of the only type-α object which witnesses the property denoted by $\lambda x_\alpha.\phi$ (if such an object exists). To avoid the *empty restriction problem*

[8]Thus, Tweedledum (Tweedledee) does not exist unless Tweedledee (Tweedledum) exists.

[9]The 'lowest' formula of this kind is $\forall i \exists x \exists y.(x = dum \wedge y = dee) \Rightarrow E(x,i) = E(y,i)$, where *dum* and *dee* are individual constants. I owe this observation to Sam Sanders.

from (**Winter, 1997**), cf. (**Winter, 2004**), we assume that the iota operator is partial (s.t. $\iota x_\alpha.\phi$ is only defined if $\exists x_\alpha.\phi \wedge (\forall y.y = (\iota x.\phi) \Rightarrow y = x)$).

In virtue of the above, the operator ι obeys axiom **Ax13**:

Ax13. $\phi\{y := (\iota x_\alpha.\phi)\}$

This axiom asserts that the result of substituting the variable y in a formula ϕ by a same-type term whose referent uniquely witnesses the property denoted by $\lambda y.\phi$ will be true.

The predicate *abt* from Definition 6.1.3, clause (vii) enables the definition of strong single-type objects (in Ch. 8). This predicate is a non-logical constant of the type $(e\,(s;t);t)$ which asserts that the proposition denoted by some type-$(s;t)$ formula (e.g. the proposition denoted by 'Fa') carries semantic information about a given individual a. As a result, we will sometimes refer to the predicate *abt* as the '*aboutness predicate*'.

Axioms **Ax14** to **Ax17** (below) ensure that the aboutness predicate *abt* identifies the *expected* aboutness subjects of a proposition (s.t. this predicate will enable a strong single-type interpretation of proper names and sentences along the lines described in Chapter 4.2.3). In the axioms, $A_1, \ldots, A_n$, and R are constants of the types e and $(\alpha_1 \ldots \alpha_n s;t)$, respectively, where $\alpha_1, \ldots, \alpha_n = e$. We assume that p and q are propositional variables of the type $(s;t)$. The variables x, y, and i are typed as above.

Ax14. $abt(A_1, \lambda i.R(A_1, \ldots, A_n, i)) \wedge \ldots \wedge abt(A_n, \lambda i.R(A_1, \ldots, A_n, i))$

Ax15. $\forall p \forall q \forall x \forall y.(abt(x,p) \wedge abt(y,q)) \Rightarrow \big((\exists i.(p \wedge q)(i) = \top) \Rightarrow \big(abt(x,(p \wedge q)) \wedge abt(y,(p \wedge q))\big)\big)$

Ax16. $\forall p \forall q \forall x.(abt(x,p) \wedge abt(x,q)) \Rightarrow abt(x,(p \vee q))$

Ax17. $\forall p \forall x.\, abt(x,p) \Rightarrow (\forall q.q = p \Rightarrow abt(x,q))$

The above axioms formalize[10] the intuitive properties of the aboutness relation from (**Perry, 1986**, p. 129).[11] Axiom **Ax14** asserts that every proposition carries information about each individual whose designating constant is a constituent of the proposition's denoting formula. Since the negation of a type-$(s;t)$ formula does not change the formula's constituent individual constants, it follows from axiom **Ax14** that the set of propositions which carry information about an individual (hereafter, the set of *aboutness-relevant propositions* w.r.t. an individual) is closed under negation. This fact is captured below:

Thm14. $\forall p \forall x.\, abt(x,p) \Rightarrow abt(x, \neg p)$

[10] Our only deviations lie in the omission of Perry's axiom (b) (our **Thm14**, which is a consequence of **Ax14**) and the addition of the condition '$\exists i.(p \wedge q)(i) = \top$' to **Ax15**.

[11] I thank John Perry for our interesting discussion of aboutness.

Axiom **Ax15** asserts that the set of aboutness-relevant propositions is closed under non-contradictory conjunction. Thus, the conjunction of two propositions will carry information about all individuals about which one of the propositional conjuncts carries information. Our restriction to *non-contradictory* conjunctions (i.e. conjunctions $(p \wedge q)$ whose designators are true at some possible situation, s.t. $(p \wedge q) \neq (\lambda i.\bot)$) is required by the robustness of the aboutness relation under semantic equivalence (cf. **Ax17**): If we would not restrict axiom **Ax15**, absurd propositions (e.g. 'Everything is self-different') – which are equivalent to the conjunction of a proposition with its complement – would carry information about every possible individual. But this contradicts our intuitions *re* aboutness.

Axiom **Ax16** asserts that the set of aboutness-relevant propositions is closed under disjunction if both disjuncts share the same aboutness subjects. As a result, the disjunction of two propositions will carry information about individuals about which *both* propositional disjuncts carry information.

Axiom **Ax17** asserts the robustness of the aboutness relation under the relation of semantic equivalence. As a result, every semantic equivalent of a proposition will carry information about each individual about which the proposition also carries information.

As a consequence of axiom **Ax14**, a single proposition will be able to carry information about more than one individual. This property of the aboutness relation is exploited in our representation of 'relational' propositions (e.g. in our representation of the proposition 'John loves Mary' from Figure 4.3). In particular, since the TY_2^3 correlate, $love_{(e\,e\,s;t)}(mary_e, john_e)$ (or Lmj), of the sentence John loves Mary contains the TY_2^3 correlates of the names John and Mary as constituents, the proposition 'John loves Mary' will carry information about both John and Mary. The existence of a proposition's different aboutness subjects is also assumed by Putnam's (**1958**) notion of *strict aboutness*, cf. (**Goodman, 1972**).

This completes our axiomatic characterization of the behavior of the predicates from Definition 6.1.3, clauses (v) to (vii). On this basis, we now return to the presentation of the TY_2^3 language L^2.

From the familiar connectives of the logic TY_2^3, the remaining TY_2^3 connectives are defined as follows. Below, i is again a TY_2^3 variable of the type s. The members of the sequence of TY_2^3 variables $\vec{x}$ have the type $\alpha_1, \ldots, \alpha_n$. The TY_2^3 constants φ and B, C have the types $(s;t)$, respectively $(\alpha_1 \ldots \alpha_n; t)$:

Notation 6.1.1. We write

$$\begin{aligned} \Box\varphi \quad &\text{for} \quad (\forall i.\varphi(i)) \\ \Diamond\varphi \quad &\text{for} \quad \neg\Box\neg\varphi \\ (B \twoheadrightarrow C) \quad &\text{for} \quad ((B \wedge C) \vee ((B \vee C) \wedge (\lambda\vec{x}.*))) = B \end{aligned}$$

The first two items from Notation 6.1.1 provide the familiar definitions of the modal box and diamond operators. The last item introduces the *approximation predicate* $\twoheadrightarrow$. This predicate (due to (**Muskens, 1995**, pp. 50, 47)) is the formal correspondent of the conditional '*if$\ldots$, *then$\ldots$' from Chapter 4.2.4 (cf. (4.2.18), (4.2.19)), which denotes the definedness relation on partial TY_2^3 objects. As a result, for the type-$(s;t)$ constants B and C, the formula $(B \twoheadrightarrow C)$ asserts that the proposition denoted by C contains the information of the proposition denoted by B, such that C is true if B is true and is false if B is false. The approximation predicate will figure in our definition of strong single-type objects from Chapter 8.

For simplicity, we will hereafter call the logical constants $*$, $\bot$, and $\Rightarrow$ *truth-functional connectives*, and refer to expressions of the type $(s;t)$ as *intensional formulas*. Truth-functional connectives distinguish themselves from the designated *single-type constants* from Definition 2.1.2. We adopt the notational conventions from Chapter 2.1. Substitution is defined as in the logic TY_0 (cf. Def. 2.1.3).

This completes our specification of TY_2^3 types and terms. We next provide a semantics for the logic TY_2^3.

6.2. Models

In line with our definition of the class of models for the logic TY_0, we first define a class of general models for the interpretation of atomic TY_2^3 terms (in Sect. 6.2.1). We then specify a class of algebraic models, which enable the interpretation of molecular TY_2^3 terms (in Sect. 6.2.2). In contrast to algebraic models of the logic TY_0, general TY_2^3 models are *logical* models, whose domains naturally exhibit an algebraic structure.

6.2.1. General Models. General models for the logic TY_2^3 are defined in analogy with general models for the logic TY_0. In particular, general TY_2^3 frames have the expected definition, where o is replaced by the types e, s, and t:

Definition 6.2.1 (General TY_2^3 frames). A *general frame* for the logic TY_2^3 is a set $F^2 = \{D_\alpha \mid \alpha \in \mathsf{2Type}\}$ of pairwise disjoint non-empty sets such that $D_e = \mathcal{A}$, $D_s = W$, $D_t = \mathbf{3} = \{\mathbf{T}, \mathbf{F}, \mathbf{N}\}$, and

$$D_{(\alpha_1 \ldots \alpha_n;\alpha_{n+1})} \subseteq \{f \mid f : (D_{\alpha_1} \times \cdots \times D_{\alpha_n}) \to D_{\alpha_{n+1}}\}$$

for all TY_2^3 types $(\alpha_1 \ldots \alpha_n;\alpha_{n+1})$, where $\alpha_{n+1} \in \{e, s, t\}$.

To ensure the axiomatizability of the TY_2^3 entailment relation (cf. Corr. 2.3), we associate complex TY_2^3 domains with subsets of function spaces. We identify $\mathcal{A}$ and W with the sets of semantically primitive individuals, respectively situations, and define $\mathbf{3}$ as the ordered set of the truth-combinations *true and not false* ($\mathbf{T}$), *false and not true* ($\mathbf{F}$), and *neither true nor false* ($\mathbf{N}$), cf. (**Belnap, 1977**). The combinations $\mathbf{T}$ and $\mathbf{F}$ correspond to the classical truth-values *true* and *false*.[12]

The combination **N** constitutes the truth-valuationally undefined element.

The partiality of the domain **3** implements the partiality requirement on single-type objects from Chapter 4.2.4. In particular, it induces a definedness order on all conjoinable-type domains. The relation of definedness is given below:

DEFINITION 6.2.2 (Definedness). Let f and g be partial functions in $D_{(\alpha_1 \ldots \alpha_n;t)}$. Then, g is *at least as well-defined as* (or is *less or equally partial than*) f, i.e. $f \sqsubseteq g$, iff, for all objects $d_1, \ldots, d_n$ of the types $\alpha_1, \ldots, \alpha_n$, if $f(d_1, \ldots, d_n) \neq \mathbf{N}$, then $f(d_1, \ldots, d_n) = g(d_1, \ldots, d_n)$.

Thus, a function g is as at least as well-defined as the function f if g assigns to the elements in $D_{\alpha_1} \times \cdots \times D_{\alpha_n}$ at least the same 'defined' values as f. As a result, the relation $\sqsubseteq$ orders TY_2^3 objects with respect to their *information content*.

We will hereafter refer to $\sqsubseteq$ as the *approximation relation*. We call the function g an *extension* of f, and f an *approximation* of g. An extension is *proper* if it holds for some element $\langle d_1, \ldots, d_n \rangle$ of $D_{\alpha_1} \times \cdots \times D_{\alpha_n}$ that $g(d_1, \ldots, d_n) \in \{\mathbf{T}, \mathbf{F}\}$ and $f(d_1, \ldots, d_n) = \mathbf{N}$. We define a *total* function as a function f which does not have a proper extension g.

In the semantics of the logic TY_2^3, the relation between L^2 terms and their associated TY_2^3 objects is established analogously to the relation between L terms and their associated TY_0 objects from Chapter 2.2.1. In particular, TY_2^3 interpretation functions I_{F^2} are defined as follows:

DEFINITION 6.2.3 (TY_2^3 Interpretation). An *interpretation function* $I_{F^2} : L^2 \to F^2$ is such that, for each non-logical constant c_α in L^2, $I_{F^2}(c_\alpha) \in D_\alpha$.

Variable assignments are also analogously defined. We define $g_{F^2}[d/x]$ by letting $g_{F^2}[d/x](x) = d$ and $g_{F^2}[d/x](y) = g_{F^2}(y)$ if $x \neq y$, and denote the set of all assignments g_{F^2} with respect to a given TY_2^3 frame F^2 by '$\mathcal{G}_{F^2}$'.

Since the logic TY_2^3 has a type for truth-values (or truth-combinations), expressions of the form $x \neq y$ qualify as terms of the logic TY_2^3. We will see below that TY_2^3 enables an *object-theoretic* characterization of its semantics.

We call the semantic value $I(c)$ (or $g(x)$) of a TY_2^3 term c (or x) the *object* denoted by c (resp. x). Depending on their type, we distinguish five different kinds of TY_2^3 objects: *individuals* and *situations* (type e, resp. s), *truth-combinations* (type t), *propositions* (type $(s;t)$), and *properties* (type $(\alpha_1 \ldots \alpha_n; t)$, with $n \geq 2$ if $\alpha_n = s$, and $n \geq 1$ otherwise). By Definition 6.2.2, only objects of the last three kinds can be partial.

[12] As a result of Belnap's abbreviation, the names '**T**' and '**F**' become ambiguous between the truth-values *true* and *false* and the truth-combinations *true and not false* (i.e. {**T**}) and *false and not true* ({**F**}). However, this ambiguity is not likely to create confusion.

On the basis of the above, we define general TY_2^3 models as follows:

DEFINITION 6.2.4 (General TY_2^3 models). A *general model* for the logic TY_2^3 is a triple $M_{F^2} = \langle F^2, I_{F^2}, V_{F^2}\rangle$, where the function $V_{F^2} : (\mathcal{G}_{F^2} \times \cup_\alpha T^2_\alpha) \to F^2$ assigns to every TY_2^3 assignment g_{F^2} and TY_2^3 term an interpretation such that

(i) $V_{F^2}(g_{F^2}, c) := I_{F^2}(c)$ if $c \in L^2$,
$V_{F^2}(g_{F^2}, x) := g_{F^2}(x)$ if $x \in \mathcal{V}^2$;

(ii) $V_{F^2}(g_{F^2}, B(A)) := \mathscr{S}^1_{(V_{F^2}(g_{F^2}, B))}\big(V_{F^2}(g_{F^2}, A)\big)$;

(iii) $V_{F^2}(g_{F^2}, \lambda x_\beta.A) := \big\{\big\langle d, V_{F^2}(g_{F^2}[d/x], A)\big\rangle \mid d \in D_\beta\big\}$;

(iv) If $\phi\{y := x\}$ and $\forall z.\phi\{y := x\} \to z = x$ holds for some x at w, then:
$V_{F^2}(g_{F^2}, \iota x_\alpha.\phi) :=$ the object d_α s.t. $V_{F^2}\big(g_{F^2}[d/x], \phi\{y := x\}\big) = \mathbf{T}$;
otherwise, $V_{F^2}(g_{F^2}, \iota x_\alpha.\phi)$ is undefined.

Above, clause (ii) uses the first slice function of the interpretation of B. Clause (iv) specifies the interpretation of iota terms according to their description from Section 6.1. Since the predicates E and *abt* are non-logical TY_2^3 constants, clause (ii) also defines the interpretation of existence- and aboutness-asserting formulas (s.t. $V_{F^2}\big(g_{F^2}, E(A, w)\big) := V_{F^2}(g_{F^2}, E)\big(V_{F^2}(g_{F^2}, A), V_{F^2}(g_{F^2}, w)\big)$).

This completes our presentation of general models for the logic TY_2^3. We next introduce a class of *algebraic* TY_2^3 models. These models enable the interpretation of the remaining complex TY_2^3 terms from Definition 6.1.3 (clauses (i), (iv)). Moreover, they satisfy the requirement of algebraicity from Chapter 4.1.1.

6.2.2. Algebraic Models. In contrast to domains of the logic TY_0 (on which we have enforced an algebraic behavior through the use of metalevel axioms), conjoinable TY_2^3 domains possess a natural algebraic structure. This is due to the fact that the set **3** is a De Morgan lattice, whose elements are partially ordered w.r.t. their degree of truth and non-falsity. As a result, the ordering relation, $\subseteq$, on **3** is described as a *logical* or *truth-ordering*. The structure $\langle \mathbf{3}, \subseteq\rangle$ (abbreviated '$\mathrm{L}_\mathbf{3}$'; in Fig. 6.1) is called a *logical* or *truth-lattice*, cf. (**Wansing and Belnap, 2010**).

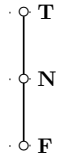

FIGURE 6.1. The logical lattice $\mathrm{L}_\mathbf{3}$.

By the truth-ordering on the set **3**, a truth-combination **X** is *contained* in the combination **Y**, i.e. $\mathbf{X} \subseteq \mathbf{Y}$, if **Y** includes truth if **X** includes truth and **X** inclu-

des falsity if **Y** includes falsity. As a result, the top and bottom elements of $\mathbf{L_3}$ are identified with the values **T** and **F**, respectively.

Since we have identified **T**, **F**, and **N** with the truth-combinations {**T**}, {**F**}, and ∅, we can analyze the meet, join, and complement operations on $\mathbf{L_3}$ as set-theoretic intersection, union, and complementation, respectively. The operations $\cap, \cup : \{\mathbf{T}, \mathbf{F}, \mathbf{N}\}^2 \to \{\mathbf{T}, \mathbf{F}, \mathbf{N}\}$ and $- : \{\mathbf{T}, \mathbf{F}, \mathbf{N}\} \to \{\mathbf{T}, \mathbf{F}, \mathbf{N}\}$ then constitute functions on the set of truth-combinations. Thus, the meet, ∩, of two truth-combinations, **X** and **Y**, is true iff **X** and **Y** are both true, and false iff at least one of **X** and **Y** is false. Dually, the join, ∪, of **X** and **Y** is true iff at least one of **X** and **Y** is true, and false iff **X** and **Y** are both false. The complement, −, of the truth-combination **X** is true iff **X** is false, and false iff **X** is true.

The above considerations define the operations ∩, ∪, and − via the truth-conditions of the Strong Kleene tables (**Kleene, 1938**), cf. (**Dunn, 1976**):

∩	**T**	**F**	**N**
T	**T**	**F**	**N**
F	**F**	**F**	**F**
N	**N**	**F**	**N**

∪	**T**	**F**	**N**
T	**T**	**T**	**T**
F	**T**	**F**	**N**
N	**T**	**N**	**N**

−	
T	**F**
F	**T**
N	**N**

TABLE 6.1. The Strong Kleene tables for ∩, ∪, and −.

The identification of the domain D_t with the truth-ordered set **3** warrants the replacement of the laws of Consistency and Excluded Middle by the complete De Morgan laws and the law of Double Negation (cf. **Ax7**, **Ax8**). In particular, since the truth-value **N** is uncomplemented (s.t. $-\mathbf{N} = \mathbf{N}$ and, thus, $(\mathbf{N} \cap -\mathbf{N}) = (\mathbf{N} \cup -\mathbf{N}) = \mathbf{N}$), the laws of Consistency and of Excluded Middle both do not obtain. The invalidity of these two laws in the structure $\langle \mathbf{3}, \cup, \cap, -, \mathbf{F}, \mathbf{T} \rangle$ motivates the characterization of algebraic TY_2^3 models as De Morgan models.

Operations on **3** are lifted on conjoinable domains by pointwise definition:

DEFINITION 6.2.5 (Lifting). Let $X = \{A_1, \ldots, A_m\}$ be the set of the type-$(\alpha_1 \ldots \alpha_n; t)$ functions $A_1, \ldots, A_m$. De Morgan operations on the members of X are defined as follows, cf. (**Gazdar, 1980**; **Groenendijk and Stokhof, 1984**): Below, $\bigcap$ and $\bigcup$ are generalized meet- and join-operations. The vector $\vec{d}$ abbreviates the sequence of objects $d_1, \ldots, d_n$ of the types $\alpha_1, \ldots, \alpha_n$.

$$
\begin{array}{lllll}
\text{(i)} & A_1 \subseteq A_2 & := & \lambda\vec{d}.A_1(\vec{d}) \subseteq A_2(\vec{d}); \\
\text{(ii)} & \bigcap X & := & \lambda\vec{d}.\bigcap\{A_i(\vec{d}) \mid A_i \in X\}; \\
\text{(iii)} & \bigcup X & := & \lambda\vec{d}.\bigcup\{A_i(\vec{d}) \mid A_i \in X\}; \\
\text{(iv)} & -A_1 & := & \lambda\vec{d}.-A_1(\vec{d}); \\
\text{(v)} & 0 & := & \lambda\vec{d}.0, \quad 1 := \lambda\vec{d}.1
\end{array}
$$

The weak definition of the generalized top and bottom elements from clause (v) (s.t. 0 and 1 are not defined via the familiar Boolean clauses $0 := \lambda d.d \cap -d$ and $1 := \lambda d.d \cup -d$) is warranted by the invalidity of LEM and the law of Consistency in the logic TY_2^3.

The lifting of $\subseteq$, $\cap$, $\cup$, $-$, 0, and 1 ensures an algebraic structure on all conjoinable-type domains. This observation is captured in Theorem 6.1:

THEOREM 6.1. *Every* TY_2^3 *domain* $D_{(\alpha_1 \ldots \alpha_n;t)}$ *forms a De Morgan algebra.*

PROOF. The proof is analogous to the proof of Theorem 2.1.

In virtue of the above, algebraic TY_2^3 models are defined through the restriction of TY_2^3 frames to conjoinable-type domains as follows:

DEFINITION 6.2.6 (Algebraic TY_2^3 models). An *algebraic* (or *De Morgan*) *model* for the logic TY_2^3 is a general model M_{F^2} for TY_2^3 with $F^2 := \bigcup_{\epsilon \in \mathsf{CoType}} D_\epsilon$, where each domain D_ϵ is the carrier of a complete De Morgan algebra.

The algebraic structure on conjoinable-type domains enables the interpretation of molecular TY_2^3 terms as follows:

DEFINITION 6.2.7. Let $M_{F^2} = \langle F^2, I_{F^2}, V_{F^2} \rangle$ be a general model for TY_2^3 and let g_{F^2} be an assignment for F^2. Then, the following holds for all B, C, and x of appropriate type:

$$
\begin{array}{lll}
\text{(i)} & V_{F^2}(g_{F^2}, (B \Rightarrow C)) & := \quad V_{F^2}(g_{F^2}, B) \subseteq V_{F^2}(g_{F^2}, C); \\
\text{(ii)} & V_{F^2}(g_{F^2}, B = C) & := \quad V_{F^2}(g_{F^2}, B) \subseteq V_{F^2}(g_{F^2}, C) \text{ and} \\
 & & \quad\;\; V_{F^2}(g_{F^2}, C) \subseteq V_{F^2}(g_{F^2}, B); \\
\text{(iii)} & V_{F^2}(g_{F^2}, \neg B) & := \quad -V_{F^2}(g_{F^2}, B); \\
\text{(iv)} & V_{F^2}(g_{F^2}, (B \wedge C)) & := \quad V_{F^2}(g_{F^2}, B) \cap V_{F^2}(g_{F^2}, C); \\
\text{(v)} & V_{F^2}(g_{F^2}, (B \vee C)) & := \quad V_{F^2}(g_{F^2}, B) \cup V_{F^2}(g_{F^2}, C); \\
\text{(vi)} & V_{F^2}(g_{F^2}, (\forall x_\alpha . \varphi)) & := \quad \bigcap_{d \in D_\alpha} V_{F^2}(g_{F^2}[d/x], \varphi); \\
\text{(vii)} & V_{F^2}(g_{F^2}, \top) & := \quad \mathbf{T}, \qquad V_{F^2}(g_{F^2}, \bot) \;:=\; \mathbf{F}, \\
 & V_{F^2}(g_{F^2}, *) & := \quad \mathbf{N}
\end{array}
$$

This completes our discussion of the interpretation of TY_2^3 terms. We next turn to the definition of truth for intensional TY_2^3 formulas.

6.3. Truth

The definition of truth for the logic TY_2^3 falls directly out of the logics' semantics. This is due to the restriction of TY_2^3 truth to intensional formulas, and the availability of designated TY_2^3 types for situations and truth-combinations. Thus, we can identify a formula's truth-value *at* a given situation with the result of applying its interpretation *to* the situation.

We call an intensional TY_2^3 formula φ *true* (or *false*) at a situation w, given a general TY_2^3 model M_{F^2} and assignment g_{F^2} if $V_{F^2}(g_{F^2}, \varphi)(w) = \mathbf{T}$ (resp. $\mathbf{F}$). We use the following abbreviation scheme, where 'M^2' abbreviates 'M_{F^2}':

NOTATION 6.3.1. We write

$$\begin{array}{lll} w \models_{M^2} \varphi & \text{for} & V_{F^2}(g_{F^2}, \varphi)(w) = \mathbf{T}; \\ w \dashv_{M^2} \varphi & \text{for} & V_{F^2}(g_{F^2}, \varphi)(w) = \mathbf{F}; \\ w \not\models_{M^2} \varphi & \text{for} & V_{F^2}(g_{F^2}, \varphi)(w) = \mathbf{N} \text{ or } \mathbf{F}; \\ w \not\dashv_{M^2} \varphi & \text{for} & V_{F^2}(g_{F^2}, \varphi)(w) = \mathbf{N} \text{ or } \mathbf{T}. \end{array}$$

As a consequence of Notation 6.3.1, it holds that

$$\begin{array}{lll} V_{F^2}(g_{F^2}, \varphi)(w) = \mathbf{T} & \text{if} & w \models_{M^2} \varphi \text{ and } w \not\dashv_{M^2} \varphi; \\ V_{F^2}(g_{F^2}, \varphi)(w) = \mathbf{F} & \text{if} & w \dashv_{M^2} \varphi \text{ and } w \not\models_{M^2} \varphi. \end{array}$$

The undefinedness of a formula's truth-value at a situation is characterized below:

$$V_{F^2}(g_{F^2}, \varphi)(w) = \mathbf{N} \quad \text{if} \quad w \not\models_{M^2} \varphi \text{ and } w \not\dashv_{M^2} \varphi.$$

Since it is possible to assign the undefined truth-value ($\mathbf{N}$), we are no longer able to identify the transmission of truth, $\models_{M^2}$, and falsity, $\dashv_{M^2}$, with the transmission of non-falsity, $\not\dashv_{M^2}$, resp. of non-truth, $\not\models_{M^2}$. This is due to the greater strictness of the modeling relations $\models_{M^2}$ and $\dashv_{M^2}$, such that the following holds:

$$\begin{array}{lll} w \not\dashv_{M^2} \varphi & \text{if} & w \models_{M^2} \varphi, (\lambda i.*) \\ w \not\models_{M^2} \varphi & \text{if} & w, (\lambda i.*) \dashv_{M^2} \varphi \end{array}$$

While the relation $w \models_{M^2} \varphi$ (or $w \dashv_{M^2} \varphi$) allows us to conclude that φ (resp. not-φ), its counterpart, $w \not\dashv_{M^2} \varphi$ (or $w \not\models_{M^2} \varphi$), only prevents us from concluding that not-φ (resp. φ).

In the logic TY_2^3, the notions of satisfiability and validity have their usual definitions:

DEFINITION 6.3.1 (TY_2^3 satisfiability). An intensional formula φ is *satisfiable* if φ is true at some situation w in some TY_2^3 model M_{F^2} under some assignment g_{F^2}, such that $w \models_{M^2} \varphi$.

DEFINITION 6.3.2 (TY_2^3 validity). An intensional formula φ is *valid* if φ is satisfiable at every situation w in every TY_2^3 model M_{F^2} under every assignment g_{F^2}, such that $\models_{M^2} \varphi$.

If the formula φ is not satisfiable, we call it *contradictory*.

We close the present section with a demonstration of the (truth-)functional completeness of the set of primitive TY_2^3 connectives. Below, we assume that the formula φ contains exactly the type-$(s;t)$ constants $p_1, \ldots, p_n$. Truth-functions for intensional TY_2^3 formulas are defined as follows:

DEFINITION 6.3.3 (Truth-functions). The function $f : \{\mathbf{T}, \mathbf{F}, \mathbf{N}\}^n \to \{\mathbf{T}, \mathbf{F}, \mathbf{N}\}$ is a truth-function for $V_{F^2}(g_{F^2}, \varphi)(w)$ if, for every general TY_2^3 model M_{F^2} and assignment g_{F^2}, $V_{F^2}(g_{F^2}, \varphi)(w) = f\big(V_{F^2}(g_{F^2}, p_1)(w), \ldots, V_{F^2}(g_{F^2}, p_n)(w)\big)$.

The following theorem establishes the expressive adequacy of the set $\{*, \bot, \Rightarrow\}$:

THEOREM 6.2 (Functional completeness). *Every truth-function is expressible by some intensional* TY_2^3 *formula.*

PROOF. The truth-functional completeness of the set $\{*, \bot, \Rightarrow\}$ follows from the completeness of the set $\{\top, \bot, \neg, \wedge, \vee, \otimes\}$ (**Blamey, 1986**, Sect. 4.1) and the definability of $\top$, $\neg$, $\wedge$, $\vee$, and $\otimes$ from $*$, $\bot$, and $\Rightarrow$, cf. (**Henkin, 1950**), where $(\varphi \otimes \psi)$ is defined as $(\varphi \wedge \psi) \vee ((\varphi \vee \psi) \wedge (\lambda i.*))$. □

This completes our truth-definition for the logic TY_2^3. We next turn to the definition of TY_2^3 entailment.

6.4. Entailment and Proof Theory

In the logic TY_2^3, entailment between type-identical terms is defined through the logical ordering on conjoinable-type domains (cf. Def. 2.4.1):

DEFINITION 6.4.1 (Generalized TY_2^3 entailment). A set of TY_2^3 terms $\Gamma = \{\gamma \mid \gamma \in T^2_{(\alpha_1 \ldots \alpha_n; t)}\}$ *entails* a set of type-identical TY_2^3 terms $\Delta = \{\delta \mid \delta \in T^2_{(\alpha_1 \ldots \alpha_n; t)}\}$, i.e. $\Gamma \models_g \Delta$, if, for all general TY_2^3 models M_{F^2} and assignments g_{F^2},

$$\bigcap_{\gamma \in \Gamma} V_{F^2}(g_{F^2}, \gamma) \subseteq \bigcup_{\delta \in \Delta} V_{F^2}(g_{F^2}, \delta)\,.$$

Definition 6.4.1 enables the following definition of semantic equivalence:

DEFINITION 6.4.2 (Global TY_2^3 equivalence). A TY_2^3 term A is (*globally*) *equivalent* to a TY_2^3 term B, i.e. $\models_g A = B$, if, for all general TY_2^3 models M_{F^2} and assignments g_{F^2}, $A \models_g B$ and $B \models_g A$.

The relation of global equivalence between terms of the logic TY_2^3 corresponds to the relation of TY_0 equivalence from Chapter 2 (Def. 2.4.2). However, to enable the identification of a name's *contextually salient* sentential equivalents in our strong single-type semantics (cf. Prop. 1.3.ii), we identify a second, weaker, notion of equivalence, called '*local equivalence*'. In the definition of local equivalence, *propositional* TY_2^3 *types* are defined as follows:

DEFINITION 6.4.3 (Propositional TY_2^3 Types). The set $\mathsf{PropType}$ of *propositional* TY_2^3 *types* is the smallest set of strings such that, for $0 \leq n \in \mathbb{N}$, if $\alpha_1, \ldots, \alpha_{n-1} \in \mathsf{2Type}$, then $(\alpha_1 \ldots \alpha_{n-1}\, s; t) \in \mathsf{PropType}$.

The local equivalence relation between TY_2^3 terms is defined below:

DEFINITION 6.4.4 (Local TY_2^3 equivalence). A propositional-type term A is *locally equivalent* to a TY_2^3 term B *at a given situation* w (or is w-*equivalent* to B), i.e. $w \models_g (\lambda\vec{x}\,\lambda i.A(\vec{x}.i) = B(\vec{x}, i))$, if $(\lambda\vec{x}.A(\vec{x}, w)) \models_g (\lambda\vec{x}.B(\vec{x}, w))$ and $(\lambda\vec{x}.B(\vec{x}, w)) \models_g (\lambda\vec{x}.A(\vec{x}, w))$ for all M_{F^2} and g_{F^2}.

In contrast to the relation of global TY_2^3 equivalence, local equivalence only demands the identity of the terms A and B at w.[13] As a result, we can define the global equivalence of two TY_2^3 terms via their local equivalence at all situations.

To enable the proof-theoretic characterization of TY_2^3 entailment, we use the connective $\Rightarrow$ from Definition 6.1.3. In Chapter 2 (cf. Thm. 2.2), we have stipulated that $\Rightarrow$ be the syntactic correspondent of the modeling relation in our single-type object theory. The existence of a TY_2^3 constant for the truth-combination **N** enables us to motivate this move: In virtue of its definition, only logical implication, $\Rightarrow$ (but not material implication, $\rightarrow$) always yields a bivalent formula. The Strong Kleene tables for $\Rightarrow$ and $\rightarrow$ are compared in Table 6.2:

$\rightarrow$	$\top$	$\bot$	$*$
$\top$	$\top$	$\bot$	$*$
$\bot$	$\top$	$\top$	$\top$
$*$	$\top$	$*$	$*$

$\Rightarrow$	$\top$	$\bot$	$*$
$\top$	$\top$	$\bot$	$\bot$
$\bot$	$\top$	$\top$	$\top$
$*$	$\top$	$\bot$	$\top$

TABLE 6.2. The Strong Kleene tables for $\rightarrow$ and $\Rightarrow$.

On the basis of the above, we establish the following deduction theorem for the logic TY_2^3. In the theorem, we let $\Gamma = \{\bigwedge_{\gamma\in\Gamma}\gamma\}$ and $\Delta = \{\bigvee_{\delta\in\Delta}\delta\}$ be sets of same-type conjoinable TY_2^3 terms.

THEOREM 6.3 (Deduction theorem for TY_2^3). *The set* Γ *entails the set* Δ *if* Δ *is deducible from* Γ:

$$\Gamma \models_g \Delta \quad \text{if} \quad \models_g \Gamma \Rightarrow \Delta$$

PROOF. The proof is standard.

We also characterize the proof-theoretic correlate of TY_2^3 entailment via a Gentzen sequent calculus. The definitions of TY_2^3 sequents and of TY_2^3 provability and refutation correspond to the definitions from Chapter 2.4. Since the structural rules for TY_2^3 are the structural rules of TY_0 (modulo the replacement of type-o by type-$(\alpha_1 \ldots \alpha_n; t)$ terms), and since the logical rules for TY_0 are analogous to the logical rules of TY_2^3 (modulo the replacement of the designated TY_0 constants by TY_2^3 connectives and quantifiers), we forego their detailed presentation.

[13] For the lack of a better alternative, we use the letter 'w' both as a situation constant and as the semantic value of this constant. However, this ambiguity is innocent.

The introduction rules for 'star' and the principle of discernibility of non-identical individuals (cf. (6.1.1)) are given in Table 6.3. In the table, i, P, and $\vec{x}$ are (sequences of) TY_2^3 variables of the types s, $(e\,s;t)$, and $\alpha_1, \ldots, \alpha_n$, respectively. The TY_2^3 constants A and c_1, c_2 have the type $(\vec{\alpha};t)$, respectively e.

$$\frac{}{(\lambda\vec{x}.*) \Rightarrow \neg(\lambda\vec{x}.*)}\,{*_1} \qquad \frac{}{\neg(\lambda\vec{x}.*) \Rightarrow (\lambda\vec{x}.*)}\,{*_2}$$

$$\frac{}{(\lambda\vec{x}.*) \Rightarrow A, \neg A}\,{*\,\mathsf{L}} \qquad \frac{}{A, \neg A \Rightarrow (\lambda\vec{x}.*)}\,{*\,\mathsf{R}}$$

$$\frac{}{c_1 \neq c_2 \Rightarrow (\exists P \exists i.P(c_1,i) \wedge (\neg P(c_2,i) \vee P(c_2,i) = *))}\,\mathsf{disc}$$

TABLE 6.3. Additional logical rules for TY_2^3.

The rules for the behavior of 'star' are due to Blamey (**1986**). In particular, the rules $*_1$ and $*_2$ are motivated by the fact that the truth-combination **N** is uncomplemented (s.t. $\mathbf{N} = \neg\mathbf{N}$). The observations that $(\mathbf{N} \vee \neg\mathbf{N}) = \mathbf{N}$ and $(\mathbf{N} \wedge \neg\mathbf{N}) = \mathbf{N}$ motivate the rules $*\,\mathsf{L}$, resp. $*\,\mathsf{R}$. In particular, the rule $*\,\mathsf{L}$ is motivated by the observation that, for every sequence $\vec{c}$ of TY_2^3 constants of the types $\alpha_1, \ldots, \alpha_n$, it holds that, if $A(\vec{c}) = \top$ or $A(\vec{c}) = \bot$, then $A(\vec{c}) \vee \neg A(\vec{c}) = \top$, such that $(\lambda\vec{x}.*)(\vec{c}) \Rightarrow (A(\vec{c}) \vee \neg A(\vec{c}))$. If $A(\vec{c}) = *$, then $(A(\vec{c}) \vee \neg A(\vec{c})) = *$, such that $(\lambda\vec{x}.*)(\vec{c}) = (A(\vec{c}) \vee \neg A(\vec{c}))$. The rule $*\,\mathsf{R}$ is motivated by the observation that, if $A(\vec{c}) = \top$ or $A(\vec{c}) = \bot$, then $A(\vec{c}) \wedge \neg A(\vec{c}) = \bot$, such that $(A(\vec{c}) \wedge \neg A(\vec{c})) \Rightarrow (\lambda\vec{x}.*)(\vec{c})$. If $A(\vec{c}) = *$, then $(A(\vec{c}) \wedge \neg A(\vec{c})) = *$, such that $(A(\vec{c}) \wedge \neg A(\vec{c})) = (\lambda\vec{x}.*)(\vec{c})$.

The metamathematical properties of TY_2^3 correspond to the properties of the logic TY_0 from Chapter 2.4.

6.5. Summary

The present chapter has introduced the three-sorted three-valued logic TY_2^3. This logic is a partial variant of Gallin's logic TY_2, whose types are formed from the basic types e, s, and t through a generalization of the type-forming rule **CT**. Terms of TY_2^3 include the symbol for logical implication, which corresponds to the logical ordering on conjoinable TY_2^3 frames. To enable the specification of semantic constraints on the PTQ translations from Definition 3.2.2, the TY_2^3 language L^2 extends the language of TY_2 with the iota operator, and with an existence and aboutness predicate. The use of these predicates will be exemplified in the following chapters.

Our focus on conjoinable types, and our assumption of the undefined truth-value **N**, ensure the partiality of certain TY_2^3 objects. In particular, the attribution of the value *undefined* to some objects induces an approximation order on con-

joinable-type objects. The existence of this ordering enables the representation of strong single-type objects along the lines described in Chapter 4.2.3.

In virtue of the above, the logic TY_2^3 constitutes the metatheory of the single-type semantics from Chapters 7 and 8. Its $(s;t)$-based subsystem WTY_1^3 is defined in the following chapter. This subsystem is obtained by restricting the sets of TY_2^3 types, terms, and frames to constructions out of the type for propositions.

CHAPTER 7

Weak Single-Type Semantics

The previous chapter has presented the metatheory of our single-type semantics in the form of the logic TY_2^3. The present chapter introduces its $(s;t)$-based subsystem WTY_1^3 and develops a weak, WTY_1^3-based, single-type semantics for Montague's PTQ fragment. This semantics is a designated model for the logic WTY_1^3 which interprets proper names, sentences, and complement phrases in the type for propositions, $(s;t)$. Objects of this type represent individuals via the set of situations in which these individuals exist, and represent propositions via the set of situations at which the propositions' denoting sentences are true. The resulting semantics enables the interpretation of all logical PTQ forms (Prop. 1.2), accommodates NP/CP complement-neutral verbs, name/CP coordinations, and CP equatives (Ch. 1.2.1) and explains the truth-evaluability of proper names (Prop. 1.3.i).

Our presentation of the TY_0-based single-type semantics from Part I has already provided a theory of syntax for the PTQ fragment (in Ch. 3.1; cf. step 1), and has formulated a set of rules for the translation of logical PTQ forms into single-type terms (in Ch. 3.2.1; cf. step 3). Consequently, to obtain a weak single-type semantics for the PTQ fragment, we only need to define the interpreting logic WTY_1^3 (step 2.i) and impose a number of semantic constraints on the interpretation of the WTY_1^3-based PTQ translations from Definition 3.2.2 (step 4). In Figure 7.1 (next page), these two steps are framed.

This chapter performs the relevant steps from the previous paragraph. In particular, Sections 7.1 to 7.2.1 and Section 7.2.2 will carry out steps (2.i) and (4), respectively. Section 7.3 specifies the truth-conditions for PTQ forms of the basic WTY_1^3 type and identifies these forms' semantic equivalents. The chapter closes with an alternative to the explanation of syntactic well-formedness from Chapter 3.4 (in Sect. 7.4). Section 7.5 contains a summary of the semantics' achievements.

7.1. The 'Weak' Object Theory WTY_1^3

We begin by defining the object theory, WTY_1^3, of the weak single-type semantics from Chapter 4.2.3. This semantics is a designated model of a subsystem of the logic TY_2^3 which constructs all of its types from the TY_2^3 type for propositions, $(s;t)$. Since the type $(s;t)$ is a propositional conjoinable TY_2^3 type, models for WTY_1^3

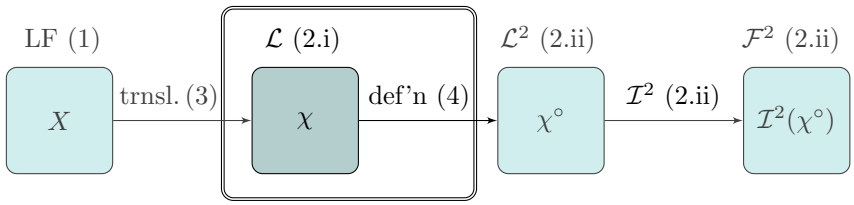

FIGURE 7.1. Doubly indirect interpretation via WTY^3_1.

will be algebraic (cf. Ch. 4.1.1), will contain partial objects (cf. Ch. 4.2.4), and will allow the truth-evaluation of basic-type terms (cf. Prop. 1.3.i).

The name of the logic, 'WTY^3_1', again follows Gallin's naming convention for type logics. In particular, the subscript '1' in its name is warranted by the construction of the least complex WTY^3_1 type, $(s;t)$, from $1+t$ basic TY^3_2 types. The letter 'W' (for '*weak*') distinguishes the presented theory from Church's Simple Theory of Types TY_1, and from the 'strong' single-type logic from Chapter 8. As for TY^3_2, the three-superscript indicates the partiality of the logic's models.

Our presentation of the logic WTY^3_1 follows the presentation of the logics TY_0 and TY^3_2. In particular, the characterization of WTY^3_1 as a subsystem of the logic TY^3_2 will allow us to skip many definitions.

7.1.1. Types and Terms. Our survey of Montagovian objects from Chapter 4.2.3 has identified functions from situations to truth-combinations (or *propositions*; cf. Partee's *properties of situations*) as the simplest 'weak' single-type candidate. As a result, we adopt the TY^3_2 type $(s;t)$ as the lowest-rank type of the logic WTY^3_1. The description of the type $(s;t)$ as the *basic*[1] WTY^3_1 type is justified by the fact that the TY^3_2 types s and t disqualify as WTY^3_1 types. Consequently, we cannot obtain the type $(s;t)$ from lower-rank WTY^3_1 types through the type-forming rule from Definition 7.1.1 (below).

The set of WTY^3_1 types is defined as a subset of the set $\mathsf{2Types}$ as follows:

DEFINITION 7.1.1 (WTY^3_1 types). The set $\mathsf{w1Type}$ of WTY^3_1 *types* is the smallest set of strings s.t., for $0 \leq n \in \mathbb{N}$, if $\alpha_1, \ldots, \alpha_n \in \mathsf{w1Type}$, then $(\alpha_1 \ldots \alpha_n\, s;t) \in \mathsf{w1Type}$.

By our basic-type choice, all types of the logic WTY^3_1 are *propositional* (cf. Def. 6.4.3) and *conjoinable* (cf. Def. 6.1.2) (s.t. $\mathsf{w1Type} \subseteq \mathsf{PropType} \subseteq \mathsf{CoType}$). As a result of the 'propositionality' of WTY^3_1 types, the linguistic correlates of basic-type

[1]The description of $(s;t)$ as the '*basic*' type of the logic WTY^3_1 is, at best, unfortunate. However, since this description supports the intuitions from Part I – and since the two uses of the adjective *basic* will be distinguished by their respective contexts –, this ambiguity is harmless.

WTY_1^3 terms will admit of a definition of truth and equivalence (cf. Sect. 7.1.4). As a result of the conjoinability of WTY_1^3 types, the logic WTY_1^3 provides natural semantic counterparts for many linguistic connectives (cf. Sect. 7.2.2).

A language L^{w1} for WTY_1^3 is a proper subset, $L^2_{\restriction \mathsf{w1Type}}$, of the set of non-logical constants of the logic TY_2^3. The set $\mathcal{V}^{w1}$ of WTY_1^3 variables is a proper subset, $\mathcal{V}^2_{\restriction \mathsf{w1Type}}$, of the set of TY_2^3 variables. From these expressions, complex terms are formed inductively via restricted variants of clauses (ii) to (iii) from Definition 6.1.3 and via type-adapted[2] variants of clauses (i) and (iv) from Definition 2.1.2:

Definition 7.1.2 (WTY_1^3 terms). Let $\alpha_1, \ldots, \alpha_n, \beta \in \mathsf{w1Type}$. The set T^{w1}_α of WTY_1^3 *terms* of the type α is then defined as follows:

(i) $L^{w1}_\alpha, \mathcal{V}^{w1}_\alpha \subseteq T^{w1}_\alpha$, $\textcircled{\scriptsize $\perp$} \in T^{w1}_{(s;t)}$;

(ii) If $\boldsymbol{B} \in T^{w1}_{(\beta\alpha_1 \ldots \alpha_n s;t)}$ and $\boldsymbol{A} \in T^{w1}_\beta$, then $(\boldsymbol{B}(\boldsymbol{A})) \in T^{w1}_{(\alpha_1 \ldots \alpha_n s;t)}$;

(iii) If $\boldsymbol{A} \in T^{w1}_{(\alpha_1 \ldots \alpha_n s;t)}$ and $\boldsymbol{x} \in \mathcal{V}_\beta$, then $(\lambda \boldsymbol{x}.\boldsymbol{A}) \in T^{w1}_{(\beta\alpha_1 \ldots \alpha_n s;t)}$;

(iv) If $\boldsymbol{B}, \boldsymbol{C} \in T^{w1}_\alpha$, then $(\boldsymbol{B} \doteRightarrow \boldsymbol{C}) \in T^{w1}_{(s;t)}$.

In clauses (i) and (iv), our use of the TY_0 symbol for the absurd entity and the TY_0 stand-in for logical implication is motivated by the ill-typing of the TY_2^3 constants $\perp$ and $\Rightarrow$ in the logic WTY_1^3, and by the availability of suitable constants in the logic TY_0. Of course, since TY_0 is an $\underline{o}$-based logic, the constants $\textcircled{\scriptsize $\perp$}_{\underline{o}}$ and $\doteRightarrow_{(\alpha\,\alpha;\underline{o})}$ are, strictly speaking, also ill-typed in WTY_1^3. However, the possibility of interpreting o as the type $(s;t)$ (Ch. 4.2.3) supports the adoption of $\textcircled{\scriptsize $\perp$}$ and $\doteRightarrow$.[3]

From $\textcircled{\scriptsize $\perp$}$ and $\doteRightarrow$, suitably typed variants of the TY_0 stand-ins for the remaining TY_2^3 connectives are obtained by an analogue of Notation 2.1.1. In particular, the constants $\textcircled{\scriptsize $\top$}$; $\boxdot$ and $\diamondsuit$; $\bigwedge$ and $\bigvee$; and $\doteq$, $\dot{\rightarrow}$, and $\dot{\leftrightarrow}$ are now non-logical constants of the types (s,t), $((s;t)\,s;t)$, $(\alpha\,(s;t)\,s;t)$, resp. $(\alpha\,\alpha\,s;t)$, where $\alpha \in \mathsf{w1Type}$. Our use of the same symbol for TY_2^3 and WTY_1^3 conjunction ($\wedge$), disjunction ($\vee$), and negation ($\neg$) is motivated by the availability of these connectives in the logic WTY_1^3, such that they have their familiar type (i.e. $(\alpha\,\alpha;\alpha)$, resp. $(\alpha;\alpha)$).[4]

We will see in Section 7.2.2 that the definition of the designated WTY_1^3 constants in terms of their associated TY_2^3 connectives constrains the semantic behavior of these constants without the use of meta-level axioms.

This completes our specification of WTY_1^3 types and terms. We next turn to the definition of general WTY_1^3 frames and models.

---

[2] In this adaptation, we replace all TY_0 types $(\alpha_1 \ldots \alpha_n; o)$ by WTY_1^3 types $(\alpha_1 \ldots \alpha_n\, s;t)$.

[3] Our bold-scripting of WTY_1^3 terms (in clauses (ii)–(iv)) and the circling of nullary TY_2^3 constants to yield their WTY_1^3 correspondents (in clause (i)) serve the same purpose.

[4] Thus, in the logic WTY_1^3, we write $\neg\boldsymbol{B}$ for $(\lambda\boldsymbol{x}.\boldsymbol{B} \doteq (\lambda\boldsymbol{y}.\textcircled{\scriptsize $\perp$}))$ and $(\boldsymbol{B} \wedge \boldsymbol{C})$ for $(\lambda\boldsymbol{x}.(\lambda\boldsymbol{X}.\boldsymbol{X}(\boldsymbol{B} \doteq \boldsymbol{C})) \doteq (\lambda\boldsymbol{X}.\boldsymbol{X}(\textcircled{\scriptsize $\top$}))))$. The connective $\vee$ then has its usual definition.

7.1.2. Models. General WTY_1^3 frames are $\mathsf{w1Type}$-restricted variants of general frames for the logic TY_2^3. These variants have the expected definition:

Definition 7.1.3 (General WTY_1^3 frames). A *general* WTY_1^3 *frame* F^{w1} is a proper subset, $F^2_{\restriction\mathsf{w1Type}} = \{D_\alpha^{F^2} \mid \alpha \in \mathsf{w1Type}\}$, of the relevant TY_2^3 frame.

Since we have identified the TY_2^3 domain $D_{(s;t)}$ with a subset of the function space $(W \to \mathbf{3})$, the ground domain of the logic WTY_1^3 contains partial objects, which are ordered with respect to their degrees of truth and definedness. As a result, WTY_1^3 frames contain the desired partial objects from Chapter 4.2.4.

Definitions 7.1.2 and 7.1.3 warrant the definition of WTY_1^3 interpretation functions $I_{F^{w1}}$ and variable assignments $g_{F^{w1}}$ as subsets of TY_2^3 interpretation functions and variable assignments (in (7.1.1), resp. (7.1.2)). General models $M_{F^{w1}}$ for the logic WTY_1^3 are then defined as subsets of general models for TY_2^3 (in (7.1.3)):

$$I_{F^2\restriction\mathsf{w1Type}} : L^2_{\restriction\mathsf{w1Type}} \to F^2_{\restriction\mathsf{w1Type}} \tag{7.1.1}$$

$$g_{F^2\restriction\mathsf{w1Type}} : \mathcal{V}^2_{\restriction\mathsf{w1Type}} \to F^2_{\restriction\mathsf{w1Type}} \tag{7.1.2}$$

$$M_{F^2\restriction\mathsf{w1Type}} = \langle F^2_{\restriction\mathsf{w1Type}}, I_{F^2\restriction\mathsf{w1Type}}, V_{F^2\restriction\mathsf{w1Type}}\rangle \tag{7.1.3}$$

Since WTY_1^3 types are conjoinable, all general models for the logic WTY_1^3 are algebraic De Morgan models (cf. Def. 6.2.6).

On the basis of the above, we next turn to the notions of truth, equivalence, and entailment for basic-type WTY_1^3 terms.

7.1.3. Truth and Entailment. In contrast to the 'pure' single-type logic from Part I, the logic WTY_1^3 commands a truth-definition for its basic-type terms. This is due to the fact that the basic WTY_1^3 type is the TY_2^3 type for propositions $(s;t)$, and that the notions of TY_2^3 truth and falsity are defined for terms of this type (cf. Nota. 6.3.1). However, since the logic WTY_1^3 does not command designated types for situations (s) or truth-combinations (t), we need to evaluate the truth or falsity of basic WTY_1^3 terms in models of the WTY_1^3 metatheory, TY_2^3.

The truth (or falsity) of basic-type WTY_1^3 terms is defined below. In the definition, an 'embedded' WTY_1^3 model $M_{F^{w1}}$ and assignment function $g_{F^{w1}}$ of a general TY_2^3 model M_{F^2} (abbr. 'M_2') and assignment g_{F^2} (abbr. 'g_2') are understood as the results of restricting (the relevant constituents of) M_2 and g_2 to WTY_1^3 terms and frames (s.t. $M_{F^{w1}} = M^{2\restriction\mathsf{w1Type}}$ and $g_{F^{w1}} = g^{2\restriction\mathsf{w1Type}}$).

The truth or falsity of basic-type WTY_1^3 terms is then defined as follows:

Definition 7.1.4 (WTY_1^3 truth). A WTY_1^3 term $\boldsymbol{A}_{(s;t)}$ is *true* (or *false*) at a situation w in an embedding TY_2^3 model, M^2, of a general WTY_1^3 model $M_{F^{w1}}$ under an embedded assignment, g^2, of the assignment $g_{F^{w1}}$ iff $w \models_{M^2} \boldsymbol{A}$ (resp. $w \neq\!|_{M^2} \boldsymbol{A}$).

Since we have defined the logic WTY_1^3 as an $(s;t)$-based subsystem of the logic TY_2^3, the relation of WTY_1^3 entailment is a restricted variant of the generalized entailment relation for terms of the logic TY_2^3 (cf. Def. 6.4.1). The relation of WTY_1^3 entailment is defined as follows:

DEFINITION 7.1.5 (WTY_1^3 entailment). A set of basic-type WTY_1^3 terms $\Gamma = \{\gamma \,|\, \gamma \in T^{w1}_{(s;t)}\}$ *entails* a set of WTY_1^3 terms $\Delta = \{\delta \,|\, \delta \in T^{w1}_{(s;t)}\}$, i.e. $\Gamma \models_g \Delta$, if, for all general WTY_1^3 models $M_{F^{w1}}$ and assignments $g_{F^{w1}}$,

$$\bigcap_{\gamma\in\Gamma} V_{F^{w1}}(g_{F^{w1}}, \gamma) \subseteq \bigcup_{\delta\in\Delta} V_{F^{w1}}(g_{F^{w1}}, \delta)\,.$$

On the basis of the above, the relation of WTY_1^3 *equivalence* is a type-$(s;t)$ restricted variant of the global equivalence relation between terms of the logic TY_2^3 (cf. Def. 6.4.2).

Like the proof theory of the single-type logic TY_0, the proof theory of the logic WTY_1^3 characterizes entailment via the TY_2^3 symbol for logical implication, $\Rightarrow$. As a result, it holds for all sets of basic WTY_1^3 terms Γ and Δ that $\Gamma \models_g \Delta$ iff $\models_g \Gamma \Rightarrow \Delta$ (Thm. 2.2, cf. Thm. 6.3). The behavior of $\Rightarrow$ is governed by WTY_1^3-typed variants of the rules of the logic TY_0 (cf. Tables 2.1–2.3).

In virtue of its proximity to the logic TY_2^3, the logic WTY_1^3 has the desired metamathematical properties from Theorems 2.3 and 2.4 and Corollaries 2.1 to 2.3. In the interest of space, we here omit their detailed statements.

This completes our discussion of the $(s;t)$-based logic WTY_1^3. We next show that a designated model of this logic interprets the PTQ fragment (cf. Prop. 1.2), accommodates the mentioned phenomena from lexical syntax, syntactic coordination, and specification (cf. Ch. 1.2.1) and explains the truth-evaluability of proper names (Prop. 1.3.i).

7.2. A WTY_1^3 Semantics for the PTQ Fragment

To identify the WTY_1^3 translations of logical PTQ forms (cf. steps 2.i, 3), we first specify the particular WTY_1^3 language $\mathcal{L}^{w1}$ and frame $\mathcal{F}^{w1}$. Their elements are (interpretations of) the designated single-type constants $\textcircled{\perp}$, $\doteq\!\!\Rightarrow$, etc. from Section 7.1.1 and the weak single-type translations, respectively interpretations of the lexical items from Table 3.1.

7.2.1. Fixing $\mathcal{L}^{w1}$, $\mathcal{F}^{w1}$, $\mathcal{L}^2$, and $\mathcal{F}^2$. Table 7.1 (next page) contains the non-logical constants of the language $\mathcal{L}^{w1}$. Table 7.2 introduces our notational conventions for WTY_1^3 variables. To ease the readability of WTY_1^3 terms – and to indicate their representational relations to terms of the logic TY_2^3 –, we again introduce different variables for type-identical objects (e.g. $\boldsymbol{x}$ and $\boldsymbol{p}$ in Table 7.2).

Constant	WTY_1^3 Type
$\boldsymbol{B}$, $\boldsymbol{C}$	$(\alpha_1 \ldots \alpha_n\, s; t)$
$\neg$	$((\alpha_1 \ldots \alpha_n\, s; t)\, \alpha_1 \ldots \alpha_n\, s; t)$
$\wedge, \vee$	$((\alpha_1 \ldots \alpha_n\, s; t)\, (\alpha_1 \ldots \alpha_n\, s; t)\, \alpha_1 \ldots \alpha_n\, s; t)$
$\bigwedge, \bigvee$	$(\alpha\, (s; t)\, s; t)$
$\dot{\Rightarrow}, \dot{=}, \dot{\neq}, \dot{\rightarrow}, \dot{\leftrightarrow}$	$(\alpha\, \alpha\, s; t)$
$\top, \bot$, ***john***, ***mary***, ***bill***, ***ninety***, ***sherlock***, ***pat***, ***moriarty***, ***partee***, ***w***	$(s; t)$
$\Box, \Diamond$, ***man***, ***woman***, ***park***, ***fish***, ***pen***, ***unicorn***, ***room***, ***problem***	$((s; t)\, s; t)$
run, ***walk***, ***talk***, ***wait***, ***arrive***, ***E***	$((s; t)\, s; t)$
find, ***lose***, ***eat***, ***love***, ***date***, ***remember***, ***fear***, ***hate***, ***destroy***, ***enter***, ***believe***, ***assert***	$((s; t)\, (s; t)\, s; t)$
temp, ***price***, ***rise***, ***change***	$(((s; t)\, s; t)\, s; t)$
rapidly, ***slowly***, ***voluntary***, ***allegedly***, ***try***, ***wish***	$(((s; t)\, s; t)\, (s; t)\, s; t)$
in, ***for***	$((s; t)\, ((s; t)\, s; t)\, (s; t)\, s; t)$
seek, ***conceive***	$((((s; t)\, s; t)\, s; t)\, (s; t)\, s; t)$
about	$((((s; t)\, s; t)\, s; t)\, ((s; t)\, s; t)\, (s; t)\, s; t)$

Table 7.1. $\mathcal{L}^{w1}$ constants.

Variable	WTY_1^3 Type
$\boldsymbol{x}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{p}, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_n, \boldsymbol{q}, \boldsymbol{r}$	$(s; t)$
$\boldsymbol{P}, \boldsymbol{P}_1, \ldots, \boldsymbol{P}_n$	$((s; t)\, s; t)$
$\boldsymbol{Q}, \boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_n$	$(((s; t)\, s; t)\, s; t)$
$\boldsymbol{L}, \boldsymbol{L}_1, \ldots, \boldsymbol{L}_n$	$((((s; t)\, s; t)\, s; t)\, (s; t)\, s; t)$

Table 7.2. WTY_1^3 variables.

We assume that, like the designated frame of the logic TY_0, the WTY_1^3 frame $\mathcal{F}^{w1}$ is very large. The designated interpretation function $\mathcal{I}_{\mathcal{F}^{w1}}$ sends all $\mathcal{L}^{w1}$ constants from Table 7.1 to their semantic values in $\mathcal{F}^{w1}$. The function $\mathcal{I}_{\mathcal{F}^{w1}}$ respects the conventional lexical relations between content words.

To constrain the interpretations of the WTY_1^3 translations of PTQ forms (as motivated in Ch. 5.3.1), we further need to specify a designated language $\mathcal{L}^2$, frame $\mathcal{F}^2$, and interpretation function $\mathcal{I}_{\mathcal{F}^2}$ of the logic TY_2^3 (cf. Ch. 6). The particular members of $\mathcal{L}^2$ and the relevant TY_2^3 variables are listed in Tables 7.3 and 7.4.

Since we want to define WTY_1^3 terms through expressions of the logic TY_2^3, we let the non-logical TY_2^3 constants and variables from Tables 7.3 and 7.4 include the WTY_1^3 constants and variables from Tables 7.1 and 7.2 (cf. the last entry in

CONSTANT	TY$_2^3$ TYPE
john, *mary*, *bill*, *ninety*, *sherlock*, *moriarty*, *pat*, *partee*, c	e
@	s
φ, ψ	$(s;t)$
man, *woman*, *park*, *fish*, *pen*, *unicorn*, *problem*, *room*	$(e\,s;t)$
run, *walk*, *talk*, *wait*, *arrive*, E	$(e\,s;t)$
find, *lose*, *eat*, *love*, *date*, *remember*, *fear*, *destroy*, *hate*, *enter*	$(e\,e\,s;t)$
believe, *assert*	$((s;t)\,e\,s;t)$
temp, *price*, *rise*, *change*	$(((s;t);e)\,s;t)$
rapidly, *slowly*, *voluntary*, *allegedly*, *try*, *wish*	$((e\,s;t)\,e\,s;t)$
in, *for*	$(e\,(e\,s;t)\,e\,s;t)$
seek, *conceive*	$(((e\,s;t)\,s;t)\,e\,s;t)$
about	$(((e\,s;t)\,s;t)\,(e\,s;t)\,e\,s;t)$
All WTY$_1^3$ constants from Table 7.1 are designated TY$_2^3$ constants.	

TABLE 7.3. $\mathcal{L}^2$ constants.

VARIABLE	TY$_2^3$ TYPE	OBJECT
$i, j, k, k_1, \ldots, k_n$	s	situation
$x, x_1, \ldots, x_n, y, z$	e	individual
$p, p_1, \ldots, p_n, q, r$	$(s;t)$	proposition
$P, P_1, \ldots, P_n$	$(e\,s;t)$	1$^{\text{st}}$-order intensional indiv. p'ty
$T, T_1, \ldots, T_n$	$((s;t);e)$	proposition-to-individual-function
$Q, Q_1, \ldots, Q_n$	$((e\,s;t)\,s;t)$	2$^{\text{nd}}$-order intensional indiv. p'ty
$L, L_1, \ldots, L_n$	$((((e\,s;t)\,s;t)\,s;t)\,s;t)$	4$^{\text{th}}$-order intensional indiv. p'ty
The WTY$_1^3$ variables from Table 7.2 are in the set of TY$_2^3$ variables.		

TABLE 7.4. TY$_2^3$ variables.

Tables 7.3, resp. 7.4). To enable the truth-evaluation of basic-type WTY$_1^3$ terms in a model of the logic TY$_2^3$, we further require that the designated TY$_2^3$ frame $\mathcal{F}^2$ and interpretation function $\mathcal{I}_{\mathcal{F}^2}$ embed the designated frame and interpretation function of the logic WTY$_1^3$, such that $\mathcal{F}^{w1} = \mathcal{F}^{2\restriction \mathsf{w1Type}}$ and $\mathcal{I}_{\mathcal{F}^{w1}} = \mathcal{I}_{\mathcal{F}^2\restriction \mathsf{w1Type}}$.

7.2.2. Translations of Logical PTQ Forms. On the basis of the above, we are able to provide WTY$_1^3$ translations for lexical elements of the PTQ fragment and for the example sentences from Chapter 1.2.1. These translations are WTY$_1^3$-typed variants of the TY$_0$ translations from Definition 3.2.2. For convenience, we already include the translations of the linguistic connectives from Definition 3.2.4.

DEFINITION 7.2.1 (Basic WTY_1^3 translations). The rule (T0) translates the lexical elements from Table 3.1 into the following WTY_1^3 terms, where $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$, $\boldsymbol{R}$, and $\boldsymbol{R}_1$ are WTY_1^3 variables of the types $\alpha_1, \ldots, \alpha_n$, resp. $(\alpha_1 \ldots \alpha_n\, s; t)$:

John	$\rightsquigarrow$	$\boldsymbol{john}$;	Mary	$\rightsquigarrow$	$\boldsymbol{mary}$;
Bill	$\rightsquigarrow$	$\boldsymbol{bill}$;	ninety	$\rightsquigarrow$	$\boldsymbol{ninety}$;
Pat	$\rightsquigarrow$	$\boldsymbol{pat}$;	B. Partee	$\rightsquigarrow$	$\boldsymbol{partee}$;
Moriarty	$\rightsquigarrow$	$\boldsymbol{moriarty}$;	Sherlock	$\rightsquigarrow$	$\boldsymbol{sherlock}$;
man	$\rightsquigarrow$	$\boldsymbol{man}$;	who(m)/which	$\rightsquigarrow$	$\lambda \boldsymbol{P}.\boldsymbol{P}$;
fish	$\rightsquigarrow$	$\boldsymbol{fish}$;	woman	$\rightsquigarrow$	$\boldsymbol{woman}$;
park	$\rightsquigarrow$	$\boldsymbol{park}$;	unicorn	$\rightsquigarrow$	$\boldsymbol{unicorn}$;
pen	$\rightsquigarrow$	$\boldsymbol{pen}$;	temperature	$\rightsquigarrow$	$\boldsymbol{temp}$;
price	$\rightsquigarrow$	$\boldsymbol{price}$;	fears	$\rightsquigarrow$	$\lambda \boldsymbol{Q} \lambda \boldsymbol{x}.\boldsymbol{Q}(\lambda \boldsymbol{y}.\boldsymbol{fear}(\boldsymbol{y}, \boldsymbol{x}))$;
problem	$\rightsquigarrow$	$\boldsymbol{problem}$;	hates	$\rightsquigarrow$	$\lambda \boldsymbol{Q} \lambda \boldsymbol{x}.\boldsymbol{Q}(\lambda \boldsymbol{y}.\boldsymbol{hate}(\boldsymbol{y}, \boldsymbol{x}))$;
room	$\rightsquigarrow$	$\boldsymbol{room}$;	finds	$\rightsquigarrow$	$\lambda \boldsymbol{Q} \lambda \boldsymbol{x}.\boldsymbol{Q}(\lambda \boldsymbol{y}.\boldsymbol{find}(\boldsymbol{y}, \boldsymbol{x}))$;
waits	$\rightsquigarrow$	$\boldsymbol{wait}$;	loses	$\rightsquigarrow$	$\lambda \boldsymbol{Q} \lambda \boldsymbol{x}.\boldsymbol{Q}(\lambda \boldsymbol{y}.\boldsymbol{lose}(\boldsymbol{y}, \boldsymbol{x}))$;
arrives	$\rightsquigarrow$	$\boldsymbol{arrive}$;	eats	$\rightsquigarrow$	$\lambda \boldsymbol{Q} \lambda \boldsymbol{x}.\boldsymbol{Q}(\lambda \boldsymbol{y}.\boldsymbol{eat}(\boldsymbol{y}, \boldsymbol{x}))$;
runs	$\rightsquigarrow$	$\boldsymbol{run}$;	loves	$\rightsquigarrow$	$\lambda \boldsymbol{Q} \lambda \boldsymbol{x}.\boldsymbol{Q}(\lambda \boldsymbol{y}.\boldsymbol{love}(\boldsymbol{y}, \boldsymbol{x}))$;
walks	$\rightsquigarrow$	$\boldsymbol{walk}$;	dates	$\rightsquigarrow$	$\lambda \boldsymbol{Q} \lambda \boldsymbol{x}.\boldsymbol{Q}(\lambda \boldsymbol{y}.\boldsymbol{date}(\boldsymbol{y}, \boldsymbol{x}))$;
talks	$\rightsquigarrow$	$\boldsymbol{talk}$;	destroys	$\rightsquigarrow$	$\lambda \boldsymbol{Q} \lambda \boldsymbol{x}.\boldsymbol{Q}(\lambda \boldsymbol{y}.\boldsymbol{destroy}(\boldsymbol{y}, \boldsymbol{x}))$;
exists	$\rightsquigarrow$	$\boldsymbol{E}$;	remembers	$\rightsquigarrow$	$\lambda \boldsymbol{Q} \lambda \boldsymbol{x}.\boldsymbol{Q}(\lambda \boldsymbol{y}.\boldsymbol{remember}(\boldsymbol{y}, \boldsymbol{x}))$;
rises	$\rightsquigarrow$	$\boldsymbol{rise}$;	enters	$\rightsquigarrow$	$\lambda \boldsymbol{Q} \lambda \boldsymbol{x}.\boldsymbol{Q}(\lambda \boldsymbol{y}.\boldsymbol{enter}(\boldsymbol{y}, \boldsymbol{x}))$;
changes	$\rightsquigarrow$	$\boldsymbol{change}$;	is	$\rightsquigarrow$	$\lambda \boldsymbol{Q} \lambda \boldsymbol{x}.\boldsymbol{Q}(\lambda \boldsymbol{y}.\boldsymbol{x} \doteq \boldsymbol{y})$;
seeks	$\rightsquigarrow$	$\boldsymbol{seek}$;	believes	$\rightsquigarrow$	$\lambda \boldsymbol{p} \lambda \boldsymbol{Q}.\lambda \boldsymbol{Q}(\lambda \boldsymbol{x}.\boldsymbol{believe}(\boldsymbol{p}, \boldsymbol{x}))$;
conceives	$\rightsquigarrow$	$\boldsymbol{conceive}$;	asserts	$\rightsquigarrow$	$\lambda \boldsymbol{p} \lambda \boldsymbol{Q}.\lambda \boldsymbol{Q}(\lambda \boldsymbol{x}.\boldsymbol{assert}(\boldsymbol{p}, \boldsymbol{x})) \wedge \boldsymbol{p}$;
about	$\rightsquigarrow$	$\boldsymbol{about}$;	in	$\rightsquigarrow$	$\lambda \boldsymbol{Q} \lambda \boldsymbol{P} \lambda \boldsymbol{x}.\boldsymbol{Q}(\lambda \boldsymbol{y}.\boldsymbol{in}(\boldsymbol{y}, \boldsymbol{P}, \boldsymbol{x})$;
allegedly	$\rightsquigarrow$	$\boldsymbol{allegedly}$;	for	$\rightsquigarrow$	$\lambda \boldsymbol{Q} \lambda \boldsymbol{P} \lambda \boldsymbol{x}.\boldsymbol{Q}(\lambda \boldsymbol{y}.\boldsymbol{for}(\boldsymbol{y}, \boldsymbol{P}, \boldsymbol{x})$;
that	$\rightsquigarrow$	$\lambda \boldsymbol{p}.\boldsymbol{p}$;	slowly	$\rightsquigarrow$	$\lambda \boldsymbol{P} \lambda \boldsymbol{x}.\boldsymbol{slowly}(\boldsymbol{P}, \boldsymbol{x}) \wedge \boldsymbol{P}(\boldsymbol{x})$;
tries to	$\rightsquigarrow$	$\boldsymbol{try}$;	rapidly	$\rightsquigarrow$	$\lambda \boldsymbol{P} \lambda \boldsymbol{x}.\boldsymbol{rapidly}(\boldsymbol{P}, \boldsymbol{x}) \wedge \boldsymbol{P}(\boldsymbol{x})$;
wishes to	$\rightsquigarrow$	$\boldsymbol{wish}$;	voluntarily	$\rightsquigarrow$	$\lambda \boldsymbol{P} \lambda \boldsymbol{x}.\boldsymbol{voluntary}(\boldsymbol{P}, \boldsymbol{x}) \wedge \boldsymbol{P}(\boldsymbol{x})$;
possibly	$\rightsquigarrow$	$\lambda \boldsymbol{p}.\Diamond \boldsymbol{p}$;	necessarily	$\rightsquigarrow$	$\lambda \boldsymbol{p}.\Box \boldsymbol{p}$;
t_n/it$_n$	$\rightsquigarrow$	$\boldsymbol{x}_n$, for each n;	a	$\rightsquigarrow$	$\lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \bigvee \boldsymbol{x}.\boldsymbol{P}_1(\boldsymbol{x}) \wedge \boldsymbol{P}(\boldsymbol{x})$;
(s)he$_n$	$\rightsquigarrow$	$\boldsymbol{x}_n$, for each n;	every	$\rightsquigarrow$	$\lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \bigwedge \boldsymbol{x}.\boldsymbol{P}_1(\boldsymbol{x}) \dot{\rightarrow} \boldsymbol{P}(\boldsymbol{x})$;
the	$\rightsquigarrow$	$\lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}.(\boldsymbol{P}_1(\boldsymbol{y}) \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}) \wedge \boldsymbol{P}(\boldsymbol{x})$;			
and	$\rightsquigarrow$	$\lambda \boldsymbol{R}_1 \lambda \boldsymbol{R} \lambda \vec{\boldsymbol{X}}.\boldsymbol{R}(\vec{\boldsymbol{X}}) \wedge \boldsymbol{R}_1(\vec{\boldsymbol{X}})$;			
or	$\rightsquigarrow$	$\lambda \boldsymbol{R}_1 \lambda \boldsymbol{R} \lambda \vec{\boldsymbol{X}}.\boldsymbol{R}(\vec{\boldsymbol{X}}) \vee \boldsymbol{R}_1(\vec{\boldsymbol{X}})$;			
not	$\rightsquigarrow$	$\lambda \boldsymbol{R} \lambda \vec{\boldsymbol{X}}.\neg \boldsymbol{R}(\vec{\boldsymbol{X}})$			

To identify the particular TY_2^3 referent of every WTY_1^3 term (cf. Ch. 5.3), we constrain the interpretation of the primitive WTY_1^3 constants from Table 7.1. From these constraints, the constraints for the remaining WTY_1^3 terms from Definition 7.2.1 can then be obtained via a compositional definition.

DEFINITION 7.2.2 (Definition of $\mathcal{L}^{w1}$-constants). The interpretations of the WTY$_1^3$ constants from Table 7.1 obey the following semantic constraints: In (C9)–(C13), we let X abbreviate $\iota P.(\forall \boldsymbol{z} \forall k_1.\boldsymbol{P}(\boldsymbol{z}, k_1) = P([\iota z.\boldsymbol{z} = (\lambda k_2.E(z, k_2))], k_1))$.

(C1) $\textcircled{\perp} = \lambda i.\perp$

(C2) $(\boldsymbol{B} \Rightarrow \boldsymbol{C}) = \lambda i.\boldsymbol{B}(i) \Rightarrow \boldsymbol{C}(i)$

(C0) $\boldsymbol{w} = \lambda i \forall p.p(@) \twoheadrightarrow p(i)$

(C3)
$\boldsymbol{john} = \lambda i.E(john, i)$; $\boldsymbol{sherlock} = \lambda i.E(sherlock, i)$;
$\boldsymbol{mary} = \lambda i.E(mary, i)$; $\boldsymbol{moriarty} = \lambda i.E(moriarty, i)$;
$\boldsymbol{bill} = \lambda i.E(bill, i)$; $\boldsymbol{pat} = \lambda i.E(pat, i)$;
$\boldsymbol{ninety} = \lambda i.E(ninety, i)$; $\boldsymbol{partee} = \lambda i.E(partee, i)$

(C4)
$\boldsymbol{man} = \lambda \boldsymbol{x} \lambda i.man\big([\iota x.\boldsymbol{x} = (\lambda j.E(x, j))], i\big)$;
$\boldsymbol{woman} = \lambda \boldsymbol{x} \lambda i.woman\big([\iota x.\boldsymbol{x} = (\lambda j.E(x, j))], i\big)$;
$\boldsymbol{park} = \lambda \boldsymbol{x} \lambda i.park\big([\iota x.\boldsymbol{x} = (\lambda j.E(x, j))], i\big)$;
$\boldsymbol{fish} = \lambda \boldsymbol{x} \lambda i.fish\big([\iota x.\boldsymbol{x} = (\lambda j.E(x, j))], i\big)$;
$\boldsymbol{pen} = \lambda \boldsymbol{x} \lambda i.pen\big([\iota x.\boldsymbol{x} = (\lambda j.E(x, j))], i\big)$;
$\boldsymbol{unicorn} = \lambda \boldsymbol{x} \lambda i.unicorn\big([\iota x.\boldsymbol{x} = (\lambda j.E(x, j))], i\big)$;
$\boldsymbol{problem} = \lambda \boldsymbol{x} \lambda i.problem\big([\iota x.\boldsymbol{x} = (\lambda j.E(x, j))], i\big)$;
$\boldsymbol{room} = \lambda \boldsymbol{x} \lambda i.room\big([\iota x.\boldsymbol{x} = (\lambda j.E(x, j))], i\big)$

(C5)
$\boldsymbol{run} = \lambda \boldsymbol{x} \lambda i.run\big([\iota x.\boldsymbol{x} = (\lambda j.E(x, j))], i\big)$;
$\boldsymbol{walk} = \lambda \boldsymbol{x} \lambda i.walk\big([\iota x.\boldsymbol{x} = (\lambda j.E(x, j))], i\big)$;
$\boldsymbol{talk} = \lambda \boldsymbol{x} \lambda i.talk\big([\iota x.\boldsymbol{x} = (\lambda j.E(x, j))], i\big)$;
$\boldsymbol{wait} = \lambda \boldsymbol{x} \lambda i.wait\big([\iota x.\boldsymbol{x} = (\lambda j.E(x, j))], i\big)$;
$\boldsymbol{arrive} = \lambda \boldsymbol{x} \lambda i.arrive\big([\iota x.\boldsymbol{x} = (\lambda j.E(x, j))], i\big)$;
$\boldsymbol{E} = \lambda \boldsymbol{x} \lambda i.E\big([\iota x.\boldsymbol{x} = (\lambda j.E(x, j))], i\big)$

(C6)
$\boldsymbol{believe} = \lambda \boldsymbol{p} \lambda \boldsymbol{x} \lambda i.believe\big(\boldsymbol{p}, [\iota x.\boldsymbol{x} = (\lambda j.E(x, j))], i\big)$;
$\boldsymbol{assert} = \lambda \boldsymbol{p} \lambda \boldsymbol{x} \lambda i.assert\big(\boldsymbol{p}, [\iota x.\boldsymbol{x} = (\lambda j.E(x, j))], i\big)$

(C7)
$\boldsymbol{find} = \lambda \boldsymbol{y} \lambda \boldsymbol{x} \lambda i.find\big([\iota y.\boldsymbol{y} = (\lambda j.E(y, j))], [\iota x.\boldsymbol{x} = (\lambda k.E(x, k))], i\big)$;
$\boldsymbol{lose} = \lambda \boldsymbol{y} \lambda \boldsymbol{x} \lambda i.lose\big([\iota y.\boldsymbol{y} = (\lambda j.E(y, j))], [\iota x.\boldsymbol{x} = (\lambda k.E(x, k))], i\big)$;
$\boldsymbol{eat} = \lambda \boldsymbol{y} \lambda \boldsymbol{x} \lambda i.eat\big([\iota y.\boldsymbol{y} = (\lambda j.E(y, j))], [\iota x.\boldsymbol{x} = (\lambda k.E(x, k))], i\big)$;
$\boldsymbol{love} = \lambda \boldsymbol{y} \lambda \boldsymbol{x} \lambda i.love\big([\iota y.\boldsymbol{y} = (\lambda j.E(y, j))], [\iota x.\boldsymbol{x} = (\lambda k.E(x, k))], i\big)$;
$\boldsymbol{date} = \lambda \boldsymbol{y} \lambda \boldsymbol{x} \lambda i.date\big([\iota y.\boldsymbol{y} = (\lambda j.E(y, j))], [\iota x.\boldsymbol{x} = (\lambda k.E(x, k))], i\big)$;
$\boldsymbol{fear} = \lambda \boldsymbol{y} \lambda \boldsymbol{x} \lambda i.fear\big([\iota y.\boldsymbol{y} = (\lambda j.E(y, j))], [\iota x.\boldsymbol{x} = (\lambda k.E(x, k))], i\big)$;
$\boldsymbol{hate} = \lambda \boldsymbol{y} \lambda \boldsymbol{x} \lambda i.hate\big([\iota y.\boldsymbol{y} = (\lambda j.E(y, j))], [\iota x.\boldsymbol{x} = (\lambda k.E(x, k))], i\big)$;
$\boldsymbol{destroy} = \lambda \boldsymbol{y} \lambda \boldsymbol{x} \lambda i.destroy\big([\iota y.\boldsymbol{y} = (\lambda j.E(y, j))], [\iota x.\boldsymbol{x} = (\lambda k.E(x, k))], i\big)$;
$\boldsymbol{enter} = \lambda \boldsymbol{y} \lambda \boldsymbol{x} \lambda i.enter\big([\iota y.\boldsymbol{y} = (\lambda j.E(y, j))], [\iota x.\boldsymbol{x} = (\lambda k.E(x, k))], i\big)$;
$\boldsymbol{remember} = \lambda \boldsymbol{y} \lambda \boldsymbol{x} \lambda i.remember\big([\iota y.\boldsymbol{y} = (\lambda j.E(y, j))], [\iota x.\boldsymbol{x} = (\lambda k.E(x, k))], i\big)$

$$
\begin{array}{lll}
\text{(C8)} & \boldsymbol{temp} & = \lambda \boldsymbol{P}\lambda i.\mathit{temp}\big(\big[\iota T.\big(\forall \boldsymbol{z}.\boldsymbol{P}(\boldsymbol{z}) = [\lambda j.E(T(\boldsymbol{z}),j)]\big)\big],i\big);\\
& \boldsymbol{price} & = \lambda \boldsymbol{P}\lambda i.\mathit{price}\big(\big[\iota T.\big(\forall \boldsymbol{z}.\boldsymbol{P}(\boldsymbol{z}) = [\lambda j.E(T(\boldsymbol{z}),j)]\big)\big],i\big);\\
& \boldsymbol{rise} & = \lambda \boldsymbol{P}\lambda i.\mathit{rise}\big(\big[\iota T.\big(\forall \boldsymbol{z}.\boldsymbol{P}(\boldsymbol{z}) = [\lambda j.E(T(\boldsymbol{z}),j)]\big)\big],i\big);\\
& \boldsymbol{change} & = \lambda \boldsymbol{P}\lambda i.\mathit{change}\big(\big[\iota T.\big(\forall \boldsymbol{z}.\boldsymbol{P}(\boldsymbol{z}) = [\lambda j.E(T(\boldsymbol{z}),j)]\big)\big],i\big)\\
\text{(C9)} & \boldsymbol{rapidly} & = \lambda \boldsymbol{P}\lambda \boldsymbol{x}\lambda i.\mathit{rapidly}\big(X,[\iota x.\boldsymbol{x} = (\lambda j.E(x,j))],i\big);\\
& \boldsymbol{slowly} & = \lambda \boldsymbol{P}\lambda \boldsymbol{x}\lambda i.\mathit{slowly}\big(X,[\iota x.\boldsymbol{x} = (\lambda j.E(x,j))],i\big);\\
& \boldsymbol{voluntary} & = \lambda \boldsymbol{P}\lambda \boldsymbol{x}\lambda i.\mathit{voluntary}\big(X,[\iota x.\boldsymbol{x} = (\lambda j.E(x,j))],i\big);\\
& \boldsymbol{allegedly} & = \lambda \boldsymbol{P}\lambda \boldsymbol{x}\lambda i.\mathit{allegedly}\big(X,[\iota x.\boldsymbol{x} = (\lambda j.E(x,j))],i\big)\\
\text{(C10)} & \boldsymbol{try} & = \lambda \boldsymbol{P}\lambda \boldsymbol{x}\lambda i.\mathit{try}\big(X,[\iota x.\boldsymbol{x} = (\lambda k.E(x,k))],i\big);\\
& \boldsymbol{wish} & = \lambda \boldsymbol{P}\lambda \boldsymbol{x}\lambda i.\mathit{wish}\big(X,[\iota x.\boldsymbol{x} = (\lambda k.E(x,k))],i\big)\\
\text{(C11)} & \boldsymbol{in} & = \lambda \boldsymbol{y}\lambda \boldsymbol{P}\lambda \boldsymbol{x}\lambda i.\mathit{in}\big([\iota y.\boldsymbol{y} = (\lambda j.E(y,j))],X,\\
& & \qquad [\iota x.\boldsymbol{x} = (\lambda k.E(x,k))],i\big);\\
& \boldsymbol{for} & = \lambda \boldsymbol{y}\lambda \boldsymbol{P}\lambda \boldsymbol{x}\lambda i.\mathit{for}\big([\iota y.\boldsymbol{y} = (\lambda j.E(y,j))],X,\\
& & \qquad [\iota x.\boldsymbol{x} = (\lambda k.E(x,k))],i\big)\\
\text{(C12)} & \boldsymbol{seek} & = \lambda \boldsymbol{Q}\lambda \boldsymbol{x}\lambda i.\mathit{seek}\big([\iota Q.(\forall \boldsymbol{P}.(\lambda k.\boldsymbol{Q}(\boldsymbol{P},k)) = (\lambda k_3.Q(X,k_3)))],\\
& & \qquad [\iota x.\boldsymbol{x} = (\lambda j.E(x,j))],i\big);\\
& \boldsymbol{conceive} & = \lambda \boldsymbol{Q}\lambda \boldsymbol{x}\lambda i.\mathit{conceive}\big([\iota Q.(\forall \boldsymbol{P}.(\lambda k.\boldsymbol{Q}(\boldsymbol{P},k)) = (\lambda k_3.Q(X,k_3)))],\\
& & \qquad [\iota x.\boldsymbol{x} = (\lambda j.E(x,j))],i\big)\\
\text{(C13)} & \boldsymbol{about} & = \lambda \boldsymbol{Q}\lambda \boldsymbol{P}\lambda \boldsymbol{x}\lambda i.\mathit{about}\big([\iota Q.(\forall \boldsymbol{P}.(\lambda k.\boldsymbol{Q}(\boldsymbol{P},k)) = (\lambda k_3.Q(X,k_3)))],\\
& & \qquad X,[\iota x.\boldsymbol{x} = (\lambda j.E(x,j))],i\big)
\end{array}
$$

The constraints (C1) and (C2) define the designated WTY_1^3 constants $\textcircled{\scriptsize\perp}$ and $\doteq\!\!\Rightarrow$ as the results of lifting the TY_2^3 connectives $\perp$ and $\Rightarrow$ to constructions out of the basic WTY_1^3 type $(s;t)$.[5] For example, (C1) defines the constant $\textcircled{\scriptsize\perp}$ as the designator of the constant function from situations to 'bottom'.

The constraint (C0) defines the basic-type WTY_1^3 constant $\boldsymbol{w}$ as the designator of the set of all situations which extend the semantic information of the current situation, @. Thus, $\boldsymbol{w}$ denotes the propositional representation of @. This representation is needed for the definition of the WTY_1^3 translations of the sentences The temperature is ninety and Ninety rises (cf. (7.2.25), (D.1.1)).

From (C1), (C2), and Notation 2.1.1, the remaining designated WTY_1^3 constants are easily defined:

NOTATION 7.2.1. We write

$$
\begin{array}{lll}
\textcircled{\scriptsize\top} & \text{for} & (\lambda i.\top)\\
(\bigwedge \boldsymbol{x}.\boldsymbol{A}) & \text{for} & (\lambda i\forall \boldsymbol{x}.\boldsymbol{A}(i)) \qquad (\bigvee \boldsymbol{x}.\boldsymbol{A}) \quad \text{for} \quad (\lambda i\exists \boldsymbol{x}.\boldsymbol{A}(i))
\end{array}
$$

[5] This is similar to the translation of dynamic to typed terms from (**Muskens, 1991**, p. 9).

$$\begin{array}{llll} \boldsymbol{B} \doteq \boldsymbol{C} & \text{for} & (\lambda \boldsymbol{x} \lambda i.\, \boldsymbol{B}(\boldsymbol{x}, i) = \boldsymbol{C}(\boldsymbol{x}, i)) & \neg \boldsymbol{B} \quad \text{for} \quad (\lambda \boldsymbol{x} \lambda i.\, \neg \boldsymbol{B}(\boldsymbol{x}, i)) \\ (\boldsymbol{B} \wedge \boldsymbol{C}) & \text{for} & (\lambda \boldsymbol{x} \lambda i.\, \boldsymbol{B}(\boldsymbol{x}, i) \wedge \boldsymbol{C}(\boldsymbol{x}, i)) & \boxdot \boldsymbol{A} \quad \text{for} \quad (\lambda i.\, \boxdot \boldsymbol{A}) \\ (\boldsymbol{B} \vee \boldsymbol{C}) & \text{for} & (\lambda \boldsymbol{x} \lambda i.\, \boldsymbol{B}(\boldsymbol{x}, i) \vee \boldsymbol{C}(\boldsymbol{x}, i)) & \diamondsuit \boldsymbol{A} \quad \text{for} \quad (\lambda i.\, \diamondsuit \boldsymbol{A}) \end{array}$$

For example, the definition of the WTY_1^3 stand-in, $\top$, for *verum* (previous page) is obtained by using the TY_2^3 definitions of the WTY_1^3 constants $\bot$ and $\dot{\Rightarrow}$ in the definition of $\top$, and by replacing the resulting TY_2^3 term by its β-equivalent:

$$\begin{array}{lll} (7.2.1) \qquad \top & = & (\bot \dot{\Rightarrow} \bot) \\ & = & (\lambda i.(\lambda j.\bot)(i) \Rightarrow (\lambda j.\bot)(i)) \\ & = & (\lambda i.\bot \Rightarrow \bot) \;\; = \;\; (\lambda i.\top) \end{array}$$

The definition of the WTY_1^3 stand-in, $\boxdot$, for the modal box operator relies on the type-$(s;t)$ representation of situations w along the lines of (C0) (cf. Ch. 4.2.4; in (7.2.2), line 6), and on our restriction of situations to *possible* situations (s.t. $\neg \exists i \forall p.p(i) = \bot$; line 5). The definition of $\boxdot$ is then obtained as follows:

$$\begin{array}{lll} (7.2.2) \quad \boxdot \boldsymbol{A} & = & \big(\bigwedge \boldsymbol{p}.\boldsymbol{p} \doteq \bot \vee (\boldsymbol{p} \dot{\Rightarrow} \boldsymbol{A}) \big) \\ & = & \big(\bigwedge \boldsymbol{p} \lambda j.(\lambda k.\boldsymbol{p}(k) = \bot)(j) \vee (\lambda k_1.\boldsymbol{p}(k_1) \Rightarrow \boldsymbol{A}(k_1))(j) \big) \\ & = & \big(\lambda i \forall \boldsymbol{p}.[\lambda j.\boldsymbol{p}(j) = \bot \vee (\boldsymbol{p}(j) \Rightarrow \boldsymbol{A}(j))](i) \big) \\ & = & \big(\lambda i \forall \boldsymbol{p}.\boldsymbol{p}(i) = \bot \vee (\boldsymbol{p}(i) \Rightarrow \boldsymbol{A}(i)) \big) \\ & = & \big(\lambda i \forall \boldsymbol{p}.\boldsymbol{p}(i) \Rightarrow \boldsymbol{A}(i) \big) \\ & = & \big(\lambda i \forall \boldsymbol{p}.\boldsymbol{p}(i) = [\exists w_s \forall q.q(w) \twoheadrightarrow q(i)] \Rightarrow \boldsymbol{A}(i) \big) \\ & = & (\lambda i \forall j.\boldsymbol{A}(j)) \quad = \quad (\lambda i.\boxdot \boldsymbol{A}) \end{array}$$

The definition of $\diamondsuit$ is analogously obtained. The derivations of the TY_2^3 definitions of the remaining designated constants from Notation 7.2.1 are included in Appendix C.3. The constants $\dot{\neq}$, $\dot{\rightarrow}$, and $\dot{\leftrightarrow}$ have their expected definitions.

The η-equivalence of conjunctive, disjunctive, and negative WTY_1^3 terms to their definitions in the logic TY_2^3 obviates the need for designated WTY_1^3 connectives for conjunction, disjunction, or negation (cf. Sect. 7.1.1).

We next turn to the semantic constraints (i.e. (C3)–(C13)) for the definition of WTY_1^3 constants which do not serve as proxies for logical or situation-denoting TY_2^3 constants. We start with a discussion of the constraint (C3).

In line with the type-$(s;t)$ representation of individuals from Chapter 4.2.3 (cf. (4.2.10)), the third instance of the constraint (C3) defines the WTY_1^3 constant ***bill*** as the designator of a function which sends situations to the truth-value of the proposition 'Bill exists' at those situations (i.e. as the designator of the characteristic function of the set of situations in which Bill exists).

The remaining constraints ensure that the WTY_1^3 translations of sentential PTQ forms are interpreted as objects of the form of (4.2.9). In particular, since the type-$((s;t)\,s;t)$ term $\boldsymbol{walk}$ is defined as the designator of a function from propositions $\boldsymbol{x}$ to the characteristic function of the set of situations at which the weak type-e correlate, $\iota x.\boldsymbol{x} = (\lambda j.E(x,j))$, of $\boldsymbol{x}$ walks (cf. (C5)), we can define the WTY_1^3 translation of the sentence Bill walks as follows:

(7.2.3)
1. $[_{\mathrm{NP}}\mathsf{Bill}] \rightsquigarrow \boldsymbol{bill} = \lambda i.E(bill,i)$
2. $[_{\mathrm{VP}}[_{\mathrm{IV}}\mathsf{walks}]] \rightsquigarrow \boldsymbol{walk} = \lambda\boldsymbol{x}\,\lambda i.\,walk\big([\iota x.\boldsymbol{x} = (\lambda j.E(x,j))],i\big)$
3. $[_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{Bill}][_{\mathrm{VP}}[_{\mathrm{IV}}\mathsf{walks}]]] \rightsquigarrow \boldsymbol{walk}(\boldsymbol{bill})$
$= \lambda\boldsymbol{x}\,\lambda i.\,walk\big([\iota x.\boldsymbol{x} = (\lambda j.E(x,j))],i\big)\,[\lambda k.E(bill,k)]$
$= \lambda i.\,walk\big([\iota x.[\lambda k.E(bill,k)] = (\lambda j.E(x,j))],i\big)$
$= \lambda i.\,walk\big([\iota x.\,bill = x],i\big)$
$= \lambda i.\,walk(bill,i)$

The step from the third to the fifth line of (7.2.3.3) is justified by the additional assumption that every type-e constant has a unique referent, and by the assumption from axiom **Ax12** (cf. Ch. 6.1). As a result, the interpretation of the term $\iota x.[\lambda k.E(bill,k)] = (\lambda j.E(x,j))$ will be defined in every model of the logic TY_2^3 which provides an interpretation for the constant *bill* (cf. Def. 6.2.4.iv).

The resulting term, $\lambda i.walk(bill,i)$, adopts the strategy for the type-$(s;t)$ representation of propositions from Chapter 4.2.3 (cf. (4.2.9)). Since it is further equivalent to Montague's translation of the sentence Bill walks from (**Montague, 1973**, p. 266), cf. (**Gallin, 1975**), the interpretations of logical PTQ forms in the designated model of the logic WTY_1^3 are not significantly more complex than their interpretation in models of Montague's Intensional Logic, IL. The small increase in semantic complexity (reflected in the larger number of required lambda conversions) is compensated for by the greater modeling power of our semantics (w.r.t. the modeling power of IL, and of the logic TY_0 from Part I). This advantage will be demonstrated in (7.2.16), (7.2.26), (7.2.34), and in Section 7.4.

From Definition 7.2.2, the definitions of the WTY_1^3 translations of all other lexical elements from Table 3.1 are easily obtained. In particular, the definition of the WTY_1^3 translation of the transitive verb finds is obtained as follows:

(7.2.4)
$\mathsf{finds} \rightsquigarrow \lambda\boldsymbol{Q}\,\lambda\boldsymbol{x}.\boldsymbol{Q}(\lambda\boldsymbol{y}.\boldsymbol{find}(\boldsymbol{y},\boldsymbol{x}))$
$= \lambda\boldsymbol{Q}\,\lambda\boldsymbol{x}.\boldsymbol{Q}\big(\lambda\boldsymbol{y}.\big[\lambda\boldsymbol{z}\,\lambda\boldsymbol{x}_1\lambda i.find\big([\iota y.\boldsymbol{z} = (\lambda j.E(y,j))],$
$[\iota x.\boldsymbol{x}_1 = (\lambda k.E(x,k))],i\big)\big](\boldsymbol{y},\boldsymbol{x})\big)$
$= \lambda\boldsymbol{Q}\,\lambda\boldsymbol{x}.\boldsymbol{Q}\big(\lambda\boldsymbol{y}\,\lambda i.find\big([\iota z.\boldsymbol{y} = (\lambda j.E(z,j))],[\iota x.\boldsymbol{x} = (\lambda k.E(x,k))],i\big)\big)$

The definitions of the WTY_1^3 translations of the determiners a, every, and the are given in items (7.2.5) to (7.2.7). These definitions involve the definitions of some of the designated WTY_1^3 constants from Notation 7.2.1. In particular, in (7.2.5) and (7.2.6) (cf. (7.2.7)), the second (resp. second-to-fourth) line uses the TY_2^3 definition of the WTY_1^3 stand-ins for the connectives $\wedge$ and $\rightarrow$ (resp. $\wedge$, $=$, and $\leftrightarrow$). The remaining lines use the TY_2^3 definitions of the WTY_1^3 stand-ins for the quantifiers $\exists$ and $\forall$:

(7.2.5)
$$\begin{aligned} \mathsf{a} &\rightsquigarrow \lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \bigvee \boldsymbol{x}.\boldsymbol{P}_1(\boldsymbol{x}) \wedge \boldsymbol{P}(\boldsymbol{x}) \\ &= \lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \bigvee \boldsymbol{x}.[\lambda j.\boldsymbol{P}_1(\boldsymbol{x},j) \wedge \boldsymbol{P}(\boldsymbol{x},j)] \\ &= \lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \lambda i \exists \boldsymbol{x}.[\lambda j.\boldsymbol{P}_1(\boldsymbol{x},j) \wedge \boldsymbol{P}(\boldsymbol{x},j)](i) \\ &= \lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \lambda i \exists \boldsymbol{x}.\boldsymbol{P}_1(\boldsymbol{x},i) \wedge \boldsymbol{P}(\boldsymbol{x},i) \end{aligned}$$

(7.2.6)
$$\begin{aligned} \mathsf{every} &\rightsquigarrow \lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \bigwedge \boldsymbol{x}.\boldsymbol{P}_1(\boldsymbol{x}) \dot{\rightarrow} \boldsymbol{P}(\boldsymbol{x}) \\ &= \lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \bigwedge \boldsymbol{x}.[\lambda j.\boldsymbol{P}_1(\boldsymbol{x},j) \rightarrow \boldsymbol{P}(\boldsymbol{x},j)] \\ &= \lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \lambda i \forall \boldsymbol{x}.[\lambda j.\boldsymbol{P}_1(\boldsymbol{x},j) \rightarrow \boldsymbol{P}(\boldsymbol{x},j)](i) \\ &= \lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \lambda i \forall \boldsymbol{x}.\boldsymbol{P}_1(\boldsymbol{x},i) \rightarrow \boldsymbol{P}(\boldsymbol{x},i) \end{aligned}$$

(7.2.7) the
$$\begin{aligned} &\rightsquigarrow \lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}.(\boldsymbol{P}_1(\boldsymbol{y}) \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}) \wedge \boldsymbol{P}(\boldsymbol{x}) \\ &= \lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}.[\lambda k.(\lambda k_2.\boldsymbol{P}_1(\boldsymbol{y},k_2) \leftrightarrow \boldsymbol{x}(k_2) = \boldsymbol{y}(k_2))(k) \wedge \boldsymbol{P}(\boldsymbol{x},k)] \\ &= \lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \lambda i \exists \boldsymbol{x}[\lambda j \forall \boldsymbol{y}.(\boldsymbol{P}_1(\boldsymbol{y},j) \leftrightarrow \boldsymbol{x}(j) = \boldsymbol{y}(j)) \wedge \boldsymbol{P}(\boldsymbol{x},j)](i) \\ &= \lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \lambda i \exists \boldsymbol{x} \forall \boldsymbol{y}.(\boldsymbol{P}_1(\boldsymbol{y},i) \leftrightarrow \boldsymbol{x}(i) = \boldsymbol{y}(i)) \wedge \boldsymbol{P}(\boldsymbol{x},i) \end{aligned}$$

To show that our constraints from Definition 7.2.2 yield the 'right' definitions of the WTY_1^3 translations of complex logical forms, we next define the TY_0 translations of the logical forms of the example sentences from Chapter 3.2. We begin with the definition of the WTY_1^3 translation of the logical form of the sentence A man walks (cf. (3.2.2)):

(7.2.8)

1. $[_{\mathrm{DET}}\mathsf{a}] \rightsquigarrow \lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \bigvee \boldsymbol{x}.\boldsymbol{P}_1(\boldsymbol{x}) \wedge \boldsymbol{P}(\boldsymbol{x})$
   $= \lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \lambda i \exists \boldsymbol{x}.\boldsymbol{P}_1(\boldsymbol{x},i) \wedge \boldsymbol{P}(\boldsymbol{x},i)$
2. $[_{\mathrm{N}}\mathsf{man}] \rightsquigarrow \boldsymbol{man} = \lambda \boldsymbol{x} \lambda i.man\big([\iota x.\boldsymbol{x} = (\lambda j.E(x,j))],i\big)$
3. $[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{man}]] \rightsquigarrow \lambda \boldsymbol{P} \bigvee \boldsymbol{x}.\boldsymbol{man}(\boldsymbol{x}) \wedge \boldsymbol{P}(\boldsymbol{x})$
   $= \lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \lambda i \exists \boldsymbol{x}.\boldsymbol{P}_1(\boldsymbol{x},i) \wedge \boldsymbol{P}(\boldsymbol{x},i)\,\big[\lambda \boldsymbol{y} \lambda k.man\big([\iota x.\boldsymbol{y} = (\lambda j.E(x,j))],k\big)\big]$
   $= \lambda \boldsymbol{P} \lambda i \exists \boldsymbol{x}.\big[\lambda \boldsymbol{y} \lambda k.man\big([\iota x.\boldsymbol{y} = (\lambda j.E(x,j))],k\big)\big](\boldsymbol{x},i) \wedge \boldsymbol{P}(\boldsymbol{x},i)$
   $= \lambda \boldsymbol{P} \lambda i \exists \boldsymbol{x}.man\big([\iota x.\boldsymbol{x} = (\lambda j.E(x,j))],i\big) \wedge \boldsymbol{P}(\boldsymbol{x},i)$

4. $[_{\text{VP}}[_{\text{IV}}\mathsf{walks}]] \rightsquigarrow \boldsymbol{walks} = \lambda\boldsymbol{x}\,\lambda i.\,walk\big([\iota x.\,\boldsymbol{x} = (\lambda j.\,E(x,j))], i\big)$

5. $[_{\text{S}}[_{\text{NP}}[_{\text{DET}}\mathsf{a}][_{\text{N}}\mathsf{man}]][_{\text{VP}}[_{\text{IV}}\mathsf{walks}]]] \rightsquigarrow \bigvee \boldsymbol{x}.\,\boldsymbol{man}(\boldsymbol{x}) \wedge \boldsymbol{walk}(\boldsymbol{x})$
$= \lambda\boldsymbol{P}\lambda i\exists\boldsymbol{x}.\,man\big([\iota x.\,\boldsymbol{x} = (\lambda j.\,E(x,j))], i\big) \wedge \boldsymbol{P}(\boldsymbol{x}, i)$
$\big[\lambda\boldsymbol{y}\,\lambda k.\,walk\big([\iota y.\,\boldsymbol{y} = (\lambda k_1.\,E(y,k_1))], k\big)\big]$
$= \lambda i\exists\boldsymbol{x}.\,man\big([\iota x.\,\boldsymbol{x} = (\lambda j.\,E(x,j))], i\big) \wedge$
$\big[\lambda\boldsymbol{y}\,\lambda k.\,walk\big([\iota y.\,\boldsymbol{y} = (\lambda k_1.\,E(y,k_1))], k\big)\big]\,(\boldsymbol{x}, i)$
$= \lambda i\exists\boldsymbol{x}.\,man\big([\iota x.\,\boldsymbol{x} = (\lambda j.\,E(x,j))], i\big) \wedge walk\big([\iota y.\,\boldsymbol{x} = (\lambda k_1.\,E(y,k_1))], i\big)$
$= \lambda i\exists x.\,man(x,i) \wedge walk(x,i)$

The definitions of the WTY_1^3 translations of the logical forms of the sentences Every man walks, The man walks, A price rises, The temperature rises, and John finds a unicorn (cf. (3.2.3)–(3.2.7)) are given below. Since the obtaining of these definitions closely follows the obtaining of the definitions from (7.2.3) and (7.2.8), we content ourselves with their statement. Some of the more interesting definitional derivations can be found in Appendix D.2.

(7.2.9) $[_{\text{S}}[_{\text{NP}}[\mathsf{every}][\mathsf{man}]][[\mathsf{walks}]]] \rightsquigarrow \bigwedge \boldsymbol{x}.\,\boldsymbol{man}(\boldsymbol{x}) \dot{\rightarrow} \boldsymbol{walk}(\boldsymbol{x})$
$= \lambda i\forall x.\,man(x,i) \rightarrow walk(x,i)$

(7.2.10) $[_{\text{S}}[_{\text{NP}}[\mathsf{the}][_{\text{N}}\mathsf{man}]][_{\text{VP}}[\mathsf{walks}]]] \rightsquigarrow \bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}.(\boldsymbol{man}(\boldsymbol{y}) \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}) \wedge \boldsymbol{walk}(\boldsymbol{x})$
$= \lambda i\exists x\forall y.(man(y,i) \leftrightarrow x = y) \wedge walk(x,i)$

(7.2.11) $[_{\text{S}}[[_{\text{DET}}\mathsf{a}][_{\text{N}}\mathsf{price}]][_{\text{VP}}[\mathsf{rises}]]] \rightsquigarrow \bigvee \boldsymbol{P}.\,\boldsymbol{price}(\boldsymbol{P}) \wedge \boldsymbol{rise}(\boldsymbol{P})$
$= \lambda i\exists T.\,price(T,i) \wedge rise(T,i)$

(7.2.12) $[_{\text{S}}[_{\text{NP}}[_{\text{DET}}\mathsf{the}][_{\text{N}}\mathsf{temperature}]][_{\text{IV}}\mathsf{rises}]]$
$\rightsquigarrow \bigvee \boldsymbol{P} \bigwedge \boldsymbol{P}_1.(\boldsymbol{temp}(\boldsymbol{P}_1) \dot{\leftrightarrow} \boldsymbol{P} \doteq \boldsymbol{P}_1) \wedge \boldsymbol{rise}(\boldsymbol{P})$
$= \lambda i\exists T\forall T_1.(temp(T_1,i) \leftrightarrow T = T_1) \wedge rise(T,i)$

(7.2.13) $[_{\text{S}}[\mathsf{John}][[\mathsf{finds}][_{\text{NP}}[\mathsf{a}][_{\text{N}}\mathsf{unicorn}]]]] \rightsquigarrow \bigvee \boldsymbol{x}.\,\boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{find}(\boldsymbol{x}, \boldsymbol{john})$
$= \lambda i\exists x.\,unicorn(x,i) \wedge find(x, john, i)$

Notably, our semantic constraints from Definition 7.2.2 still allow the use of the type-shifting functions from Definitions 3.2.3, D.1.1, and D.1.2. For the function *lift*, this is illustrated in the definition of the WTY_1^3 translation of the logical form of the sentence John finds Mary (cf. (3.2.8)):

(7.2.14)

1. $[_{\text{NP}}\textsf{Mary}] \rightsquigarrow \boldsymbol{mary} = \lambda i.E(mary, i)$
2. $[_{\text{TV}}\textsf{finds}] \rightsquigarrow \lambda\boldsymbol{Q}\,\lambda\boldsymbol{x}.\boldsymbol{Q}(\lambda\boldsymbol{y}.\boldsymbol{find}(\boldsymbol{y}, \boldsymbol{x}))$
   $= \lambda\boldsymbol{Q}\,\lambda\boldsymbol{x}.\boldsymbol{Q}\big(\lambda\boldsymbol{y}\,\lambda i.find\big([\iota z.\boldsymbol{y} = (\lambda j.E(z, j))], [\iota x.\boldsymbol{x} = (\lambda k.E(x, k))], i\big)\big)$
3. $[_{\text{VP}}[_{\text{TV}}\textsf{finds}][_{\text{NP}}\textsf{Mary}]] \rightsquigarrow \lambda\boldsymbol{x}.[lift\,(\boldsymbol{mary})](\lambda\boldsymbol{y}.\boldsymbol{find}(\boldsymbol{y}, \boldsymbol{x}))$
   $= \lambda\boldsymbol{Q}\,\lambda\boldsymbol{x}.\boldsymbol{Q}\big(\lambda\boldsymbol{y}\,\lambda i.find\big([\iota z.\boldsymbol{y} = (\lambda j.E(z, j))], [\iota x.\boldsymbol{x} = (\lambda k.E(x, k))], i\big)\big)$
   $\big[lift\,\big(\lambda k_1.E(mary, k_1)\big)\big]$
   $= \lambda\boldsymbol{Q}\,\lambda\boldsymbol{x}.\boldsymbol{Q}\big(\lambda\boldsymbol{y}\,\lambda i.find\big([\iota z.\boldsymbol{y} = (\lambda j.E(z, j))], [\iota x.\boldsymbol{x} = (\lambda k.E(x, k))], i\big)\big)$
   $\big[\lambda\boldsymbol{P}.\boldsymbol{P}\,\big(\lambda k_1.E(mary, k_1)\big)\big]$
   $= \lambda\boldsymbol{x}.\big[\lambda\boldsymbol{P}.\boldsymbol{P}\,\big(\lambda k_1.E(mary, k_1)\big)\big]\big(\lambda\boldsymbol{y}\,\lambda i.find\big([\iota z.\boldsymbol{y} = (\lambda j.E(z, j))],$
   $[\iota x.\boldsymbol{x} = (\lambda k.E(x, k))], i\big)\big)$
   $= \lambda\boldsymbol{x}.\big(\lambda\boldsymbol{y}\,\lambda i.find\big([\iota z.\boldsymbol{y} = (\lambda j.E(z, j))], [\iota x.\boldsymbol{x} = (\lambda k.E(x, k))], i\big)\big)(\lambda k_1.E(mary, k_1))$
   $= \lambda\boldsymbol{x}\,\lambda i.find\big([\iota z.(\lambda k_1.E(mary, k_1)) = (\lambda j.E(z, j))], [\iota x.\boldsymbol{x} = (\lambda k.E(x, k))], i\big)$
   $= \lambda\boldsymbol{x}\,\lambda i.find\big(mary, [\iota x.\boldsymbol{x} = (\lambda k.E(x, k))], i\big)$
4. $[_{\text{S}}[_{\text{NP}}\textsf{John}][_{\text{VP}}[_{\text{TV}}\textsf{finds}][_{\text{NP}}\textsf{Mary}]]] \rightsquigarrow \boldsymbol{find}(\boldsymbol{mary}, \boldsymbol{john})$
   $= \lambda\boldsymbol{x}\,\lambda i.find\big(mary, [\iota x.\boldsymbol{x} = (\lambda k.E(x, k))], i\big)\ [\lambda j.E(john, j)]$
   $= \lambda i.find\big(mary, john, i\big)$

The definitions of the WTY$_1^3$ translations of the sentences Pat remembers Bill and Pat remembers that Bill waits for her (cf. (3.2.9), (3.2.11)) also use the function *lift*:

(7.2.15) $[_{\text{S}}[_{\text{NP}}\textsf{Pat}][_{\text{VP}}[_{\text{TV}}\textsf{remembers}][_{\text{NP}}\textsf{Bill}]]] \rightsquigarrow \boldsymbol{remember}(\boldsymbol{bill}, \boldsymbol{pat}) = \lambda i.remember\,(bill, pat, i)$

(7.2.16)

1. $[_{\text{CP}}[_{\text{C}}\textsf{that}][_{\text{S}}[_{\text{NP}}\textsf{Bill}][_{\text{VP}}[_{\text{IV}}\textsf{waits}][_{\text{PP}}[_{\text{P}}\textsf{for}][_{\text{NP}}\textsf{she}_1]]]]] \rightsquigarrow \boldsymbol{for}(\boldsymbol{x}_1, \boldsymbol{wait}, \boldsymbol{bill})$
   $= \lambda i.for\big([\iota x_1.\boldsymbol{x}_1 = (\lambda k_2.E(x_1, k_2))], wait, bill, i\big)$
2. $[_{\text{TV}}\textsf{remembers}] \rightsquigarrow \boldsymbol{remember}$
   $= \lambda\boldsymbol{y}\,\lambda\boldsymbol{x}\,\lambda i.remember\big([\iota y.\boldsymbol{y} = (\lambda j.E(y, j))], [\iota x.\boldsymbol{x} = (\lambda k.E(x, k))], i\big)$
3. $[_{\text{VP}}[_{\text{TV}}\textsf{remembers}][_{\text{CP}}[_{\text{C}}\textsf{that}][_{\text{S}}[_{\text{NP}}\textsf{Bill}][_{\text{VP}}[_{\text{IV}}\textsf{waits}][_{\text{PP}}[_{\text{P}}\textsf{for}][_{\text{NP}}\textsf{she}_1]]]]]]$
   $\rightsquigarrow \lambda\boldsymbol{x}.\boldsymbol{remember}\big(\boldsymbol{for}(\boldsymbol{x}_1, \boldsymbol{wait}, \boldsymbol{bill}), \boldsymbol{x}\big)$
   $= \lambda\boldsymbol{y}\,\lambda\boldsymbol{x}\,\lambda i.remember\big([\iota y.\boldsymbol{y} = (\lambda j.E(y, j))], [\iota x.\boldsymbol{x} = (\lambda k.E(x, k))], i\big)$
   $\big[\lambda k.for\big([\iota x_1.\boldsymbol{x}_1 = (\lambda k_2.E(x_1, k_2))], wait, bill, k\big)\big]$

$$= \lambda\boldsymbol{x}\,\lambda i.\,\mathit{remember}\big(\underline{\big[\iota y.\,\big[\lambda k.\mathit{for}\big([\iota x_1.\boldsymbol{x}_1 = (\lambda k_2.E(x_1,k_2))],\mathit{wait},\mathit{bill},k\big)\big] = (\lambda j.E(y,j))\big]},[\iota x.\boldsymbol{x} = (\lambda k.E(x,k))],i\big)$$

4. $[_{\text{S}}\ t_1\ [_{\text{VP}}[_{\text{TV}}\textsf{remembers}][_{\text{CP}}[_{\text{C}}\textsf{that}][_{\text{S}}[_{\text{NP}}\textsf{Bill}][_{\text{VP}}[_{\text{IV}}\textsf{waits}][_{\text{PP}}[_{\text{P}}\textsf{for}][_{\text{NP}}\textsf{she}_1]]]]]]]$

$$\rightsquigarrow \boldsymbol{\mathit{remember}}\big(\boldsymbol{\mathit{for}}(\boldsymbol{x}_1,\boldsymbol{\mathit{wait}},\boldsymbol{\mathit{bill}}),\boldsymbol{x}_1\big)$$

$$= \lambda\boldsymbol{x}\,\lambda i.\,\mathit{remember}\big(\underline{\big[\iota y.\big[\lambda k.\mathit{for}\big([\iota x_1.\boldsymbol{x}_1 = (\lambda k_2.E(x_1,k_2))],\mathit{wait},\mathit{bill},k\big)\big] = (\lambda j.E(y,j))\big]},\big[\iota x.\boldsymbol{x} = (\lambda k.E(x,k))\big],i\big)$$

$$= \lambda i.\,\mathit{remember}\big(\underline{\big[\iota y.\big[\lambda k.\mathit{for}\big([\iota x_1.\boldsymbol{x}_1 = (\lambda k_2.E(x_1,k_2))],\mathit{wait},\mathit{bill},k\big)\big] = (\lambda j.E(y,j))\big]},[\iota x.\boldsymbol{x}_1 = (\lambda k.E(x,k))],i\big)$$

5. $[_{\text{S}}[_{\text{NP}}\textsf{Pat}]^1\,[_{\text{S}}\ t_1\ [_{\text{VP}}[_{\text{TV}}\textsf{remembers}][_{\text{CP}}[_{\text{C}}\textsf{that}][_{\text{S}}[_{\text{NP}}\textsf{Bill}][_{\text{VP}}[_{\text{IV}}\textsf{waits}][_{\text{PP}}[_{\text{P}}\textsf{for}][_{\text{NP}}\textsf{she}_1]]]]]]]]$

$$\rightsquigarrow \boldsymbol{\mathit{remember}}\big(\boldsymbol{\mathit{for}}(\boldsymbol{\mathit{pat}},\boldsymbol{\mathit{wait}},\boldsymbol{\mathit{bill}}),\boldsymbol{\mathit{pat}}\big)$$

$$= \lambda\boldsymbol{x}_1\lambda i.\,\mathit{remember}\big(\underline{\big[\iota y.\big[\lambda k.\mathit{for}\big([\iota x_1.\boldsymbol{x}_1 = (\lambda k_2.E(x_1,k_2))],\mathit{wait},\mathit{bill},k\big)\big] = (\lambda j.E(y,j))\big]},[\iota x.\boldsymbol{x}_1 = (\lambda k.E(x,k))],i\big)\,[\lambda k_1.E(\mathit{pat},k_1)]$$

$$= \lambda i.\,\mathit{remember}\big(\underline{\big[\iota y.\big[\lambda k.\mathit{for}\big([\iota x_1.[\lambda k_1.E(\mathit{pat},k_1)] = (\lambda k_2.E(x_1,k_2))],\mathit{wait},\mathit{bill},k\big)\big] = (\lambda j.E(y,j))\big]},[\iota x.[\lambda k_1.E(\mathit{pat},k_1)] = (\lambda k.E(x,k))],i\big)$$

$$= \lambda i.\,\mathit{remember}\big(\underline{[\iota y.[\lambda k.\mathit{for}(\mathit{pat},\mathit{wait},\mathit{bill},k)] = (\lambda j.E(y,j))]},\mathit{pat},i\big)$$

The possibility of translating the above sentences in our WTY_1^3-based semantics establishes a variant of Proposition 3.1 for this semantics:

PROPOSITION 7.1 (NP/CP neutrality). *A weak single-type semantics enables the interpretation of* (*both guises of*) NP/CP *complement-neutral expressions.*

Note that the possibility of *defining* the WTY_1^3 translations of [[TV][CP]]-structures in a weak single-type semantics is conditional on the existence of non-Montagovian individuals which serve as type-e correlates of propositions: The TY_2^3 correlate, i.e. *remember*, of the WTY_1^3 term ***remember*** restricts its first argument to TY_2^3 terms of the type e. To satisfy the typing constraints of the relevant TY_2^3 terms, we need to identify the individual which encodes the semantic information of the propositional argument. In (7.2.16), this is achieved by identifying the unique individual which exists exactly in the situations at which the formula $\lambda k.\mathit{for}(\mathit{pat},\mathit{wait},\mathit{bill},k)$ is true (cf. the underlined TY_2^3 term in (7.2.16)). However, this individual is a 'special'[6] type-e object which is not included in the set of individuals from (**Montague, 1973**). The propositional status of such individuals explai-

[6]In (**Zalta, 1983**), such individuals are described as *abstract* objects.

ns the impossibility of modeling the logical form (7.2.16) (and, similarly, (3.2.12)) in *traditional* Montague semantics.

We will see in (7.2.26) (next page) that the existence of propositional individuals is also required for the definition of the WTY_1^3 translation of CP equatives (e.g. (3.2.22)). The definitions of the WTY_1^3 translations of the logical forms of the remaining PTQ sentences from Chapter 3.2 are included below.

As expected, our definitions of the WTY_1^3 translations of the different-scope readings of the sentences John seeks a unicorn (cf. (3.2.13), (3.2.14)), John talks about a unicorn (cf. (3.2.15), (3.2.16)), and A woman loves every man (cf. (3.2.17), (3.2.18)) are equivalents of the forms' familiar Montagovian translations. These definitions are given below:

(7.2.17) $[_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{John}][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{seeks}][_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{unicorn}]]]]$

$\rightsquigarrow \ \boldsymbol{seek}\big([\lambda \boldsymbol{P}\bigvee \boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{P}(\boldsymbol{x})], \boldsymbol{john}\big)$

$= \ \lambda i.\, seek\big([\lambda P \lambda j\, \exists x.\, unicorn(x,j) \wedge P(x,j)], john, i\big)$

(7.2.18) $[_{\mathrm{S}}[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{unicorn}]]^0\,[_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{John}][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{seeks}]\ t_0]]]$

$\rightsquigarrow \ \bigvee \boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{seek}\big([\lambda \boldsymbol{P}.\boldsymbol{P}(\boldsymbol{x})], \boldsymbol{john}\big)$

$= \ \lambda i \exists x.\, unicorn(x,i) \wedge seek\big([\lambda P \lambda j.\, P(x,j)], john, i\big)$

(7.2.19) $[_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{John}][_{\mathrm{VP}}[_{\mathrm{IV}}\mathsf{talks}][_{\mathrm{PP}}[_{\mathrm{P}}\mathsf{about}][_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{unicorn}]]]]]$

$\rightsquigarrow \ \boldsymbol{about}\big([\lambda \boldsymbol{P}\bigvee \boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{P}(\boldsymbol{x})], \boldsymbol{talk}, \boldsymbol{john}\big)$

$= \ \lambda i.\, about\big([\lambda P \lambda j\, \exists x.\, unicorn(x,j) \wedge P(x,j)], talk, john, i\big)$

(7.2.20) $[_{\mathrm{S}}[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{unicorn}]]^0\,[_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{John}][_{\mathrm{VP}}[_{\mathrm{IV}}\mathsf{talks}][_{\mathrm{PP}}[_{\mathrm{P}}\mathsf{about}]\ t_0]]]]$

$\rightsquigarrow \ \bigvee \boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{about}\big([\lambda \boldsymbol{P}.\boldsymbol{P}(\boldsymbol{x})], \boldsymbol{talk}, \boldsymbol{john}\big)$

$= \ \lambda i \exists x.\, unicorn(x,i) \wedge about\big([\lambda P \lambda j.\, P(x,j)], talk, john, i\big)$

(7.2.21) $[_{\mathrm{S}}[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{woman}]][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{loves}][_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{every}][_{\mathrm{N}}\mathsf{man}]]]]$

$\rightsquigarrow \ \bigvee \boldsymbol{x}.\boldsymbol{woman}(\boldsymbol{x}) \wedge \big(\bigwedge \boldsymbol{y}.\boldsymbol{man}(\boldsymbol{y}) \dot{\rightarrow} \boldsymbol{love}(\boldsymbol{y},\boldsymbol{x})\big)$

$= \ \lambda i \exists x.\, woman(x,i) \wedge \big(\forall y.\, man(y,i) \rightarrow love(y,x,i)\big)$

(7.2.22) $[[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{every}][_{\mathrm{N}}\mathsf{man}]]^1[_{\mathrm{S}}[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{woman}]][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{loves}]\ t_1]]]$

$\rightsquigarrow \ \bigwedge \boldsymbol{y}.\boldsymbol{man}(\boldsymbol{y}) \dot{\rightarrow} \big(\bigvee \boldsymbol{x}.\boldsymbol{woman}(\boldsymbol{x}) \wedge \boldsymbol{love}(\boldsymbol{y},\boldsymbol{x})\big)$

$= \ \lambda i \forall y.\, man(y,i) \rightarrow \big(\exists x.\, woman(x,i) \wedge love(y,x,i)\big)$

The definition of the WTY_1^3 equality sign (cf. Nota. 7.2.1) constrains the interpretation of the translation of the English copula be. The WTY_1^3 translations of the LFs of its involving sentences from (3.2.19) to (3.2.21) are defined as follows:

(7.2.23) $[_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{Bill}][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{is}][_{\mathrm{NP}}\mathsf{Mary}]]] \rightsquigarrow \boldsymbol{bill} \doteq \boldsymbol{mary}$
$= \lambda i.\, bill = mary$

(7.2.24) $[_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{Bill}][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{is}][_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{man}]]]] \rightsquigarrow \boldsymbol{man}(\boldsymbol{bill})$
$= (\lambda i \exists x.\, man(x,i) \wedge bill = x) \;=\; (\lambda i.\, man(bill, i))$

(7.2.25) $[_{\mathrm{S}}[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{the}][_{\mathrm{N}}\mathsf{temperature}]][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{is}][_{\mathrm{NP}}\mathsf{ninety}]]]$
$\rightsquigarrow \bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}.\big(\big(\bigvee \boldsymbol{P}_2.\, \boldsymbol{temp}(\boldsymbol{P}_2) \wedge \boldsymbol{y} \doteq \boldsymbol{P}_2(\boldsymbol{w})\big) \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}\big) \wedge \boldsymbol{x} \doteq \boldsymbol{ninety}$
$= \lambda i \exists x \forall y.\big(\big(\exists T.\, temp(T,i) \wedge y = T(\boldsymbol{w})\big) \leftrightarrow x = y\big) \wedge x = ninety$

Notably, our assumption of type-e correlates of propositions (cf. (7.2.16)) also enables the definition of the WTY_1^3 translation of the equative The problem is that Mary hates Bill (cf. (3.2.22)). This definition is obtained below:

(7.2.26)

1. $[_{\mathrm{CP}}[_{\mathrm{C}}\mathsf{that}][_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{Mary}][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{hates}][_{\mathrm{NP}}\mathsf{Bill}]]]] \rightsquigarrow \boldsymbol{hate}(\boldsymbol{bill}, \boldsymbol{mary})$
   $= \lambda i.\, hate(bill, mary, i)$
2. $[_{\mathrm{TV}}\mathsf{is}] \rightsquigarrow \lambda \boldsymbol{Q} \lambda \boldsymbol{x}.\, \boldsymbol{Q}(\lambda \boldsymbol{y}.\, \boldsymbol{x} \doteq \boldsymbol{y}) = \lambda \boldsymbol{Q} \lambda \boldsymbol{x}.\, \boldsymbol{Q}(\lambda \boldsymbol{y} \lambda i.\, \boldsymbol{x}(i) = \boldsymbol{y}(i))$
3. $[_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{is}][_{\mathrm{CP}}[_{\mathrm{C}}\mathsf{that}][_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{Mary}][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{hates}][_{\mathrm{NP}}\mathsf{Bill}]]]]]$
   $\rightsquigarrow \lambda \boldsymbol{x}.\, \boldsymbol{x} \doteq \boldsymbol{hate}(\boldsymbol{bill}, \boldsymbol{mary})$
   $= \lambda \boldsymbol{Q} \lambda \boldsymbol{x}.\, \boldsymbol{Q}(\lambda \boldsymbol{y} \lambda i.\, \boldsymbol{x}(i) = \boldsymbol{y}(i))\, [lift(\lambda j.\, hate(bill, mary, j))]$
   $= \lambda \boldsymbol{Q} \lambda \boldsymbol{x}.\, \boldsymbol{Q}(\lambda \boldsymbol{y} \lambda i.\, \boldsymbol{x}(i) = \boldsymbol{y}(i))\, [\lambda \boldsymbol{P}.\, \boldsymbol{P}(\lambda j.\, hate(bill, mary, j))]$
   $= \lambda \boldsymbol{x}.[\lambda \boldsymbol{P}.\, \boldsymbol{P}(\lambda j.\, hate(bill, mary, j))]\, (\lambda \boldsymbol{y} \lambda i.\, \boldsymbol{x}(i) = \boldsymbol{y}(i))$
   $= \lambda \boldsymbol{x}.[(\lambda \boldsymbol{y} \lambda i.\, \boldsymbol{x}(i) = \boldsymbol{y}(i))\, (\lambda j.\, hate(bill, mary, j))]$
   $= \lambda \boldsymbol{x}.[(\lambda i.\, \boldsymbol{x}(i) = (\lambda j.\, hate(bill, mary, j))(i))]$
   $= \lambda \boldsymbol{x} \lambda i.\, \boldsymbol{x}(i) = hate(bill, mary, i)$
4. $[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{the}][_{\mathrm{N}}\mathsf{problem}]] \rightsquigarrow \lambda \boldsymbol{P} \bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}.(\boldsymbol{problem}(\boldsymbol{y}) \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}) \wedge \boldsymbol{P}(\boldsymbol{x})$
   $= \lambda \boldsymbol{P} \lambda i \exists \boldsymbol{x} \forall \boldsymbol{y}.\big(problem\big([\iota x.\, \boldsymbol{y} = (\lambda j.\, E(x,j))], i\big) \leftrightarrow \boldsymbol{x} = \boldsymbol{y}\big) \wedge \boldsymbol{P}(\boldsymbol{x}, i)$
5. $[_{\mathrm{S}}[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{the}][_{\mathrm{N}}\mathsf{problem}]][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{is}][_{\mathrm{CP}}[_{\mathrm{C}}\mathsf{that}][_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{Mary}][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{hates}][_{\mathrm{NP}}\mathsf{Bill}]]]]]]$
   $\rightsquigarrow \bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}.(\boldsymbol{problem}(\boldsymbol{y}) \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}) \wedge \boldsymbol{x} \doteq \boldsymbol{hate}(\boldsymbol{bill}, \boldsymbol{mary})$
   $= \lambda \boldsymbol{P} \lambda i \exists \boldsymbol{x} \forall \boldsymbol{y}.\big(problem\big([\iota x.\, \boldsymbol{y} = (\lambda j.\, E(x,j))], i\big) \leftrightarrow \boldsymbol{x} = \boldsymbol{y}\big) \wedge \boldsymbol{P}(\boldsymbol{x}, i)$
   $[\lambda \boldsymbol{z} \lambda j.\, \boldsymbol{z}(j) = hate(bill, mary, j)]$

$$
\begin{aligned}
&= \lambda i \exists \boldsymbol{x} \forall \boldsymbol{y}. \big(\mathit{problem}\big([\iota x. \boldsymbol{y} = (\lambda j. E(x,j))], i\big) \leftrightarrow \boldsymbol{x} = \boldsymbol{y}\big) \wedge \\
&\qquad [\lambda \boldsymbol{z} \lambda j. \boldsymbol{z}(j) = \mathit{hate}(\mathit{bill}, \mathit{mary}, j)]\,(\boldsymbol{x}, i) \\
&= \lambda i \exists \boldsymbol{x} \forall \boldsymbol{y}. \big(\mathit{problem}\big([\iota x. \boldsymbol{y} = (\lambda j. E(x,j))], i\big) \leftrightarrow \boldsymbol{x} = \boldsymbol{y}\big) \wedge \boldsymbol{x}(i) = \mathit{hate}(\mathit{bill}, \mathit{mary}, i) \\
&= \lambda i \exists x \forall y. (\mathit{problem}\,(y,i) \leftrightarrow x = y) \wedge x = \underline{[\iota z. (\lambda k. \mathit{hate}(\mathit{bill}, \mathit{mary}, k)) = (\lambda j. E(z,j))]}
\end{aligned}
$$

The equivalence of the second-to-last and the last term from (7.2.26.5) is warranted by the identity of $\boldsymbol{x}$, $\boldsymbol{y}$ and $(\lambda k. \mathit{hate}(\mathit{bill}, \mathit{mary}, k))$, and by the resulting possibility of replacing $\boldsymbol{x}(i) = \mathit{hate}(\mathit{bill}, \mathit{mary}, i)$ by the term $[\iota x. \boldsymbol{x} = (\lambda j. E(x,j))] = [\iota x. (\lambda k. \mathit{hate}(\mathit{bill}, \mathit{mary}, k)) = (\lambda j. E(z,j))]$. The equivalence of $[\iota x. \boldsymbol{x} = (\lambda j. E(x,j))]$ with x yields the second conjunct from the last line of (7.2.26.5).

On the basis of the above, we next turn to the definitions of the WTY$_1^3$ translations of coordinated logical forms. These definitions involve η-expansions of the WTY$_1^3$ translations of and, or, and not from Definition 7.2.1:

$$
\begin{aligned}
&(7.2.27) \quad \lambda \boldsymbol{R}_1 \lambda \boldsymbol{R} \lambda \vec{\boldsymbol{X}}. \boldsymbol{R}(\vec{\boldsymbol{X}}) \wedge \boldsymbol{R}_1(\vec{\boldsymbol{X}}) &&= \lambda \boldsymbol{R}_1 \lambda \boldsymbol{R} \lambda \vec{\boldsymbol{X}} \lambda i. \boldsymbol{R}(\vec{\boldsymbol{X}}, i) \wedge \boldsymbol{R}_1(\vec{\boldsymbol{X}}, i) \\
&(7.2.28) \quad \lambda \boldsymbol{R}_1 \lambda \boldsymbol{R} \lambda \vec{\boldsymbol{X}}. \boldsymbol{R}(\vec{\boldsymbol{X}}) \vee \boldsymbol{R}_1(\vec{\boldsymbol{X}}) &&= \lambda \boldsymbol{R}_1 \lambda \boldsymbol{R} \lambda \vec{\boldsymbol{X}} \lambda i. \boldsymbol{R}(\vec{\boldsymbol{X}}, i) \vee \boldsymbol{R}_1(\vec{\boldsymbol{X}}, i) \\
&(7.2.29) \quad \lambda \boldsymbol{R} \lambda \vec{\boldsymbol{X}}. \neg \boldsymbol{R}(\vec{\boldsymbol{X}}) &&= \lambda \boldsymbol{R} \lambda \vec{\boldsymbol{X}} \lambda i. \neg \boldsymbol{R}(\vec{\boldsymbol{X}}, i)
\end{aligned}
$$

By using these definitions, we can define the WTY$_1^3$ translations of the sentences Bill walks and every man finds a unicorn (cf. (3.2.23)), John finds and eats a unicorn (cf. (3.2.24)), and John does not find a unicorn (cf. (3.2.27)):

$$
\begin{aligned}
&(7.2.30) \quad [_{\mathrm{S}}[_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{Bill}][_{\mathrm{VP}}[_{\mathrm{IV}}\mathsf{walks}]]] \\
&\qquad\qquad [[_{\mathrm{CONJ}}\mathsf{and}][_{\mathrm{S}}[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{every}][_{\mathrm{N}}\mathsf{man}]][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{finds}][_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{unicorn}]]]]]] \\
&\qquad \rightsquigarrow \boldsymbol{walk}(\boldsymbol{bill}) \wedge \big(\bigwedge \boldsymbol{y}. \boldsymbol{man}(\boldsymbol{y}) \dot{\rightarrow} \big(\bigvee \boldsymbol{x}. \boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{find}(\boldsymbol{y}, \boldsymbol{x}) \big) \big) \\
&\qquad = \lambda i. walk(bill, i) \wedge \big(\forall y. man(y, i) \rightarrow \big(\exists x. unicorn(x, i) \wedge find(y, x, i) \big) \big)
\end{aligned}
$$

$$
\begin{aligned}
&(7.2.31) \quad [_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{John}][_{\mathrm{VP}}[_{\mathrm{TV}}[_{\mathrm{TV}}\mathsf{finds}][[_{\mathrm{CONJ}}\mathsf{and}][_{\mathrm{TV}}\mathsf{eats}]]][_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{unicorn}]]]] \\
&\qquad \rightsquigarrow \bigvee \boldsymbol{x}. \boldsymbol{unicorn}(\boldsymbol{x}) \wedge (\boldsymbol{find} \wedge \boldsymbol{eat})(\boldsymbol{x}, \boldsymbol{john}) \\
&\qquad = \lambda i \exists x. unicorn(x, i) \wedge (find \wedge eat)(x, john, i)
\end{aligned}
$$

$$
\begin{aligned}
&(7.2.32) \quad [_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{John}][_{\mathrm{VP}}[_{\mathrm{ADV}}\mathsf{does\ not}][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{find}][_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{unicorn}]]]]] \\
&\qquad \rightsquigarrow \neg \big(\bigvee \boldsymbol{x}. \boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{find}(\boldsymbol{x}, \boldsymbol{john}) \big) \\
&\qquad = \lambda i. \neg \big(\exists x. unicorn(x, i) \wedge find(x, john, i) \big)
\end{aligned}
$$

Remark that our definition of the WTY$_1^3$ translation of the verb exists enables a *consistent* interpretation of negative existential sentences:[7]

(7.2.33)

1. $[_{\text{NP}}\textsf{John}] \rightsquigarrow \boldsymbol{john} = \lambda i.E(john, i)$
2. $[_{\text{VP}}[_{\text{IV}}\textsf{exist}]] \rightsquigarrow \boldsymbol{E} = \lambda\boldsymbol{x}\lambda i.E\big([\iota x.\boldsymbol{x} = (\lambda j.E(x, j))], i\big)$
3. $[\textsf{does not}] \rightsquigarrow (\lambda\boldsymbol{P}\lambda\boldsymbol{x}.\neg\boldsymbol{P}(\boldsymbol{x})) = (\lambda\boldsymbol{P}\lambda\boldsymbol{x}\lambda i.\neg\boldsymbol{P}(\boldsymbol{x}, i))$
4. $[_{\text{VP}}[\textsf{does not}][_{\text{VP}}[_{\text{IV}}\textsf{exist}]]] \rightsquigarrow \lambda\boldsymbol{x}.\neg\boldsymbol{E}(\boldsymbol{x})$
   $= \lambda\boldsymbol{x}\lambda i.\neg E\big([\iota x.\boldsymbol{x} = (\lambda j.E(x, j))], i\big)$
5. $[_{\text{S}}[_{\text{NP}}\textsf{John}][_{\text{VP}}[\textsf{does not}][_{\text{VP}}[_{\text{IV}}\textsf{exist}]]]] \rightsquigarrow \neg\boldsymbol{E}(\boldsymbol{john})$
   $= \lambda i.\neg E\big([\iota x.[\lambda k.E(john, k)] = (\lambda j.E(x, j))], i\big)$
   $= \lambda i.\neg E(john, i)$

In virtue of their particular form, our constraints on the interpretation of WTY_1^3 constants also enable the definition of the WTY_1^3 translations of coordinations with a proper name- and a CP-conjunct. However, since the TY_2^3 correlates (i.e. *find*, *remember*) of some WTY_1^3 constants (e.g. ***find***, ***remember***) restrict their first argument to TY_2^3 terms of the type e – and since the TY_2^3 counterpart of linguistic conjunction is only defined for conjoinable types (and is, thus, not available for type-e terms) –, we are unable to define the WTY_1^3 translation of the result of coordinating a name and a CP in the complement of transitive verbs (cf. (3.2.25)). To compensate for this inability, we instead define the WTY_1^3 translation of the coordination of the results of combining the transitive verb with the NP, respectively CP (in (7.2.34)), or of the coordination of the results of combining the thus-obtained VPs with the proper name Pat (in (7.2.35)):

(7.2.34)

$[_{\text{S}}[_{\text{NP}}\textsf{Pat}]^1\,[_{\text{S}}\;t_1\,[_{\text{VP}}[_{\text{TV}}\textsf{remembers}][_{\text{NP}}\textsf{Bill}]]$
$[[_{\text{CONJ}}\textsf{and}][_{\text{VP}}[_{\text{TV}}\textsf{remembers}][_{\text{CP}}[_{\text{C}}\textsf{that}][_{\text{S}}[_{\text{NP}}\textsf{Bill}][_{\text{VP}}[_{\text{IV}}\textsf{waits}][_{\text{PP}}[_{\text{P}}\textsf{for}][_{\text{NP}}\textsf{she}_1]]]]]]]]]$

$\rightsquigarrow \big(\lambda\boldsymbol{x}.\boldsymbol{remember}(\boldsymbol{bill}, \boldsymbol{x}) \wedge \boldsymbol{remember}\big(\boldsymbol{for}(\boldsymbol{x}, \boldsymbol{wait}, \boldsymbol{bill}), \boldsymbol{x}\big)\big)(\boldsymbol{pat})$

$= \big(\lambda x\lambda i.remember(bill, x, i) \wedge$
$remember\big([\iota y.[\lambda k.for(x, wait, bill, k)] = (\lambda j.E(y, j))], x, i\big)\big)(pat)$

(7.2.35)

$[_{\text{S}}[_{\text{S}}[_{\text{NP}}\textsf{Pat}]^1[_{\text{VP}}[_{\text{TV}}\textsf{remembers}][_{\text{NP}}\textsf{Bill}]]][[_{\text{CONJ}}\textsf{and}]$
$[_{\text{S}}[_{\text{NP}}\textsf{Pat}]^1\,[_{\text{S}}\;t_1\;[_{\text{VP}}[_{\text{TV}}\textsf{remembers}][_{\text{CP}}[_{\text{C}}\textsf{that}][_{\text{S}}[_{\text{NP}}\textsf{Bill}][_{\text{VP}}[_{\text{IV}}\textsf{waits}][_{\text{PP}}[_{\text{P}}\textsf{for}][_{\text{NP}}\textsf{she}_1]]]]]]]]$

$\rightsquigarrow \boldsymbol{remember}\big(\boldsymbol{bill}, \boldsymbol{pat}\big) \wedge \boldsymbol{remember}\big(\boldsymbol{for}(\boldsymbol{pat}, \boldsymbol{wait}, \boldsymbol{bill}), \boldsymbol{pat}\big)$

$= \lambda i.remember(bill, pat, i) \wedge$
$remember\big([\iota y.[\lambda k.for(pat, wait, bill, k)] = (\lambda j.E(y, j))], pat, i\big)$

[7] I owe this observation to Dietmar Zaefferer.

Since the logical forms from (7.2.34) and (7.2.35) are intuitively equivalent to the logical form from (3.2.25), their substitution for the form from (3.2.25) does not give reason for concern. However, the impossibility of defining the WTY_1^3 translation of (3.2.25) once more illustrates the merits of a single-type semantics (here, the merits of the semantics of the logic WTY_1^3) over traditional Montague semantics (or over the semantics of the logic TY_2^3).

The possibility of defining the WTY_1^3 translation of (an intuitive equivalent of) sentence (7.2.34) is in agreement with a weak single-type version of Proposition 3.2. This version is stated below:

PROPOSITION 7.2 (NP/CP coordinability). *A weak single-type semantics can model logical forms which contain coordinations with a proper name- and a* CP*-conjunct.*

This completes our definition of the WTY_1^3 translations from Definition 7.2.2. To assess our semantics' ability to accommodate the claims from Proposition 1.3, the next section specifies the truth-conditions for PTQ expressions of the basic WTY_1^3 type, and identifies their semantic equivalents.

7.3. PTQ Truth and Equivalence

Our presentation of the logic WTY_1^3 has already established the possibility of evaluating the truth or falsity of basic-type WTY_1^3 terms in models of the metatheory TY_2^3 (cf. Def. 7.1.4). Since we know (from Sect. 7.2.2) that the WTY_1^3 translation of every logical PTQ form is defined through a term of the logic TY_2^3, we can evaluate the truth or falsity of logical PTQ forms via the truth, respectively falsity of their translations' TY_2^3 definitions.

Below, we let $\boldsymbol{A}_{(s;t)}$ be the WTY_1^3 translation of some logical form X, such that $X \rightsquigarrow \boldsymbol{A}$. We let M^2 and $M_{\mathcal{F}^{w1}}$ be the designated models of the logics TY_2^3, respectively WTY_1^3, and let g^2 and $g^{w1} = g^{2\restriction \mathsf{w1Type}}$ be their associated assignments. We assume that w is a situation in the metatheory.

The truth or falsity of logical PTQ forms is then defined as follows:

DEFINITION 7.3.1 (WTY_1^3-based linguistic truth). A logical form X is *true* (or *false*) at the situation w in M^2 under g^2, i.e. $\text{TRUE}_{M^{w1},w}(X)$ (resp. $\text{FALSE}_{M^{w1},w}(X)$), if $w \models_{M^2} \boldsymbol{A}$ (resp. $w \dashv_{M^2} \boldsymbol{A}$).

The above definition provides the expected truth-conditions for the logical forms of PTQ sentences. In particular, the truth-conditions for the sentence Barbara Partee arrives at the current situation @ are given below:

(7.3.1) $\quad \text{TRUE}_{M^{w1},@}\big([_{\text{S}}[_{\text{NP}}\text{Barbara Partee}][_{\text{VP}}[_{\text{IV}}\text{arrives}]]]\big)$

iff $\quad @ \models_{M^2} \boldsymbol{arrive}(\boldsymbol{partee}) \quad$ iff $\quad @ \models_{M^2} \lambda i.\, arrive(partee, i)$

$$\begin{array}{lll} & \textsc{false}_{M^{w1},@}\big([_{\text{S}}[_{\text{NP}}\textsf{Barbara Partee}][_{\text{VP}}[_{\text{IV}}\textsf{arrives}]]]\big) & \\ \text{iff} & @ \nvDash_{M^2} \boldsymbol{arrive}(\boldsymbol{partee}) & \text{iff} \quad @ \nvDash_{M^2} \lambda i.\, arrive(partee, i) \\ \text{iff} & @ \models_{M^2} \neg\boldsymbol{arrive}(\boldsymbol{partee}) & \text{iff} \quad @ \models_{M^2} \lambda i.\, \neg arrive(partee, i) \end{array}$$

For example, if @ is the context from (9) – which is inhabited by Barbara Partee and in which she has the property of arriving –, the sentence Barbara Partee arrives is true at @. If @ is a *variant* of the context from (9) which is inhabited by Barbara Partee, but in which she does *not* have the property of arriving (instead, the new arrival is Angelika Kratzer), the sentence Barbara Partee arrives is false at @. If @ is the context from (10) (which is only inhabited by Mia, Rob, and their mothers, such that Barbara Partee neither has nor does not have the property of arriving), the truth-value of the sentence Barbara Partee arrives is undefined at @.

Since we interpret proper names in the basic WTY_1^3 type, Definition 7.3.1 extends the definition of linguistic truth and falsity to proper names. Thus, the name Barbara Partee is true at the current situation @ iff the definition, $\lambda i.\, E(partee, i)$, of the name's WTY_1^3 translation, $\boldsymbol{partee}$, is true at @ in M^2 under g^2, i.e. if Barbara Partee is an inhabitant of @:

$$\begin{array}{lll} (7.3.2) & \textsc{true}_{M^{w1},@}\big([_{\text{NP}}\textsf{Barbara Partee}]\big) & \\ & \text{iff} \quad @ \models_{M^2} \boldsymbol{partee} & \text{iff} \quad @ \models_{M^2} \lambda i.\, E(partee, i) \end{array}$$

The bivalent behavior of the TY_2^3 existence predicate E (cf. Ch. 6.1, **Ax10**) ensures the 'classical' truth-evaluation of names in a WTY_1^3-based semantics. Thus, the name Barbara Partee is false at @ if it is not the case that the name Barbara Partee is true at @, i.e. if Barbara Partee is *not* among the inhabitants of @:

$$\begin{array}{lll} (7.3.3) & \textsc{false}_{M^{w1},@}\big([_{\text{NP}}\textsf{Barbara Partee}]\big) & \\ & \text{iff} \quad @ \not\models_{M^2} \boldsymbol{partee} & \text{iff} \quad @ \not\models_{M^2} \lambda i.\, E(partee, i) \end{array}$$

The truth-evaluability of proper names (cf. Prop. 1.3.i) and the classicality of their WTY_1^3 truth-conditions are captured below:

Proposition 7.3 (WTY_1^3 truth-aptness of names). *In a weak single-type semantics, proper names receive a classical truth-value at every situation.*

Admittedly, the possibility of evaluating the truth of proper names in a single-type semantics again depends on the availability of a multi-typed metatheory (here, on the availability of a designated model of TY_2^3). However, since this requirement does not impact the possibility of interpreting all logical PTQ forms into constructions out of the single basic type, it is compatible with Partee's conjecture. Analogous observations hold for the requirement on the identification of equivalence relations between names and sentences (cf. Def. 7.3.2, next page).

The truth-aptness of proper names and sentences suggests the greater suitability of our weak single-type semantics in comparison to the 'pure', TY_0-based,

semantics from Part I. This greater suitability is further supported by the fact that, in a weak single-type semantics, proper names share the truth- and falsity-conditions of their containing simple existential sentences. For example, the truth (or falsity) of the name Barbara Partee thus coincides with the truth (resp. falsity) of the sentence Barbara Partee exists at each situation:

(7.3.4)

$$\begin{aligned}&\textsc{true}_{M^{w1},@}\big([_{\text{NP}}\text{B. Partee}]\big) \text{ iff } \textsc{true}_{M^{w1},@}\big([_{\text{S}}[_{\text{NP}}\text{Barbara Partee}][_{\text{VP}}[_{\text{IV}}\text{exists}]]]\big)\\ &\text{iff } @ \models_{M^2} \boldsymbol{partee} \quad \text{iff } @ \models_{M^2} \boldsymbol{E}(\boldsymbol{partee}) \text{ iff } @ \models_{M^2} \lambda i.E(partee, i)\end{aligned}$$

$$\begin{aligned}&\textsc{false}_{M^{w1},@}\big([_{\text{NP}}\text{B. Partee}]\big) \text{ iff } \textsc{false}_{M^{w1},@}\big([_{\text{S}}[_{\text{NP}}\text{Barbara Partee}][_{\text{VP}}[_{\text{IV}}\text{exists}]]]\big)\\ &\text{iff } @ \not\models_{M^2} \boldsymbol{partee} \quad \text{iff } @ \not\models_{M^2} \boldsymbol{E}(\boldsymbol{partee}) \text{ iff } @ \not\models_{M^2} \lambda i.E(partee, i)\end{aligned}$$

In virtue of the above, our WTY_1^3-based semantics enables the identification of *some* of a name's equivalent sentences. This observation is captured below:

Proposition 7.4 (Existential WTY_1^3 equivalents of names). *In a weak single-type semantics, proper names stand in equivalence relations to the logical forms of some existential sentences. Instances of this relation are identifiable via the global equivalence of the* TY_2^3 *definitions of the forms'* WTY_1^3 *translations.*

The semantic equivalence of logical PTQ forms is defined below. In this definition, we let $\boldsymbol{A}_{(s;t)}$ and $\boldsymbol{B}_{(s;t)}$ be the WTY_1^3 translations of the logical forms X, respectively Y, such that $X \rightsquigarrow \boldsymbol{A}$ and $Y \rightsquigarrow \boldsymbol{B}$.

Definition 7.3.2 (WTY_1^3-based linguistic equivalence). The form X is semantically *equivalent* to Y in M^{w1} under g^{w1}, i.e. $\textsc{means}_{M^{w1}}(Y, X)$, if $\models_g \boldsymbol{A} = \boldsymbol{B}$ in M^2 under g^2.

The above definition improves upon the equivalence-definition of our 'pure' single-type semantics (cf. Def. 3.3.1) by enabling the identification of equivalence relations between proper names and sentences. In particular, the equivalence-condition from Definition 7.3.2 supports the equivalence of names and their containing simple existential sentences. For the name Barbara Partee, this is shown below:

(7.3.5)
$$\begin{aligned}&\textsc{means}_{M^{w1}}\big([_{\text{NP}}\text{Barbara Partee}],\ [_{\text{S}}[_{\text{NP}}\text{Barbara Partee}][_{\text{VP}}[_{\text{IV}}\text{exists}]]]\big)\\ &\quad\text{iff } \models_g \boldsymbol{partee} = \boldsymbol{E}(\boldsymbol{partee})\\ &\quad\text{iff } \models_g (\lambda i.E(partee, i)) = (\lambda i.E(partee, i)) \text{ iff } \models_g \top\end{aligned}$$

However, our WTY_1^3-based semantics still fails to identify a name's *contextually salient* sentential equivalents. In particular, the definition of basic-type WTY_1^3 constants as intensional TY_2^3 formulas of the form $\lambda i.E(c, i)$ (cf. (C3)) prevents the equivalence of proper names with contingent non-existential sentences. For example, since we can imagine a situation at which only the intensional formula $\lambda i.$ $E(partee, i)$ (but not the formula $\lambda i.arrive(partee, i)$) is true, the name Barbara

Partee fails to be equivalent to the sentence Barbara Partee arrives in our WTY_1^3-based semantics:

(7.3.6) not $\textsc{means}_{M^{w1}}\left([_{\mathrm{NP}}\textsf{B. Partee}]\,,\,[_{\mathrm{S}}[_{\mathrm{NP}}\textsf{B. Partee}][_{\mathrm{VP}}[_{\mathrm{IV}}\textsf{arrives}]]]\right)$

because $\models_g \boldsymbol{partee} \neq \boldsymbol{arrive}(\boldsymbol{partee})$

because $\models_g \left(\lambda i.E\left(partee, i\right)\right) \neq \left(\lambda i.arrive\left(partee, i\right)\right)$

because $\models_g \left(\exists i.E\left(partee, i\right) = \top \wedge arrive\left(partee, i\right) = \bot\right) \vee$
$\left(\exists i.E\left(partee, i\right) = \top \wedge arrive\left(partee, i\right) = *\right)$

The informational poverty of the TY_2^3 definitions of basic-type WTY_1^3 terms suggests that a weak single-type semantics is unable to satisfy Proposition 1.3.ii. This observation is summarized in Proposition 7.5:

Proposition 7.5 (Absence of non-existential name equivalents). *In a weak single-type semantics, proper names fail to be equivalent to contextually contingent non-existential sentences.*

Given the findings in nonsentential speech from Chapter 1.2.1, the inability to identify a name's non-existential sentential equivalents is a severe problem for our WTY_1^3-based single-type semantics. We will solve this problem in the next chapter.

To show that our WTY_1^3-based semantics shares the familiar relation of logical consequence between the logical forms of PTQ sentences, we also specify this relation. The latter is defined in analogy to the relation of TY_0-based linguistic entailment from Chapter 3.3 (cf. Def. 3.3.2).

In the definition of WTY_1^3-based linguistic entailment, we let $\Xi = \{X \mid X \rightsquigarrow \gamma\}$ and $\Upsilon = \{Y \mid Y \rightsquigarrow \delta\}$ be sets of logical forms of sentences which are translated into the sets of WTY_1^3 terms $\Gamma = \{\gamma \mid \gamma \in T^{w1}_{(s;t)}\}$ and $\Delta = \{\delta \mid \delta \in T^{w1}_{(s;t)}\}$:

Definition 7.3.3 (WTY_1^3-based linguistic entailment). A set of logical forms Ξ *entails* the set of forms Υ in the designated model, M^2, of the logic TY_2^3, i.e. $\textsc{follow}_{M^{w1}}(\Upsilon, \Xi)$, if $\models_g \Gamma \Rightarrow \Delta$ in M^2 under g^2.

The above definition validates the inference from (3.3.1) and blocks the inference from (3.3.2). Since we assume the truth of the formula $\forall x \forall i.\, arrive(x, i) \Rightarrow E(x, i)$ in the designated model of our metatheory TY_2^3, the sentence Barbara Partee arrives further entails the sentential interpretation of the proper name Barbara Partee from (9) in our weak single-type semantics.

This completes our illustration of the greater modeling power of the weak, WTY_1^3-based, single-type semantics with respect to traditional Montague semantics, or with respect to the 'pure' single-type semantics from Part I. We next assess the relative explanatory power of this new semantics.

7.4. Sorting Single-Type Objects

Admittedly, by its single-type character, our WTY_1^3-based semantics inherits the explanatory challenges of the 'pure' single-type semantics from Part I (cf. Ch. 3.4). However, the TY_2^3 definition of WTY_1^3 PTQ-translations admits of a *semantic* explanation of the ill-formedness of the expressions from (16b) to (21b). This explanation exploits the fact that the WTY_1^3 translations of most English sentences[8] are not defined by a TY_2^3 formula of the form $\lambda i.E(c,i)$, where c is some individual constant. Rather, their definition will contain some type-$(e\,s;t)$ predicate P which is not equivalent to E. As a result, we can distinguish different *sorts* (called *separation subtypes*, or *subtypes*) within the single basic type. These sorts then serve as the formal basis for syntactic categories.

Figure 7.2 illustrates the correspondence between syntactic categories and WTY_1^3 subtypes. As is easy to see, WTY_1^3 subtypes almost completely regain the granularity of the semantic distinctions in the Montagovian type system.

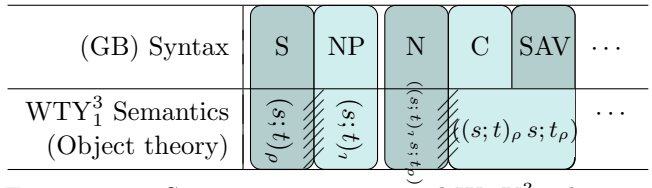

FIGURE 7.2. Syntactic categories and WTY_1^3 subtypes.

Below, we first give a formal introduction to subtypes. We then describe the use of subtypes in the formulation of syntactic well-formedness constraints.

Separation subtyping (**Lambek and Scott, 1986**), cf. (**Pollard, 2008**), is a type-forming operation which restricts a given type, α, to the set of objects (i.e. the (*separation*) *subtype* of α) satisfying some predicate $P_{(\alpha;t)}$. The resulting type is denoted by '$\{a \in \alpha \mid P(a)\}$', or '$\alpha_P$'. In particular, the separation of the basic WTY_1^3 type by the TY_2^3 predicates for the representation of individuals (in (7.4.1), next page; cf. (4.2.10)) and propositions (in (7.4.2); cf. (4.2.9)) enables an emulation of the distinction between individuals and propositions within the single basic type $(s;t)$.[9] Below, the type-$(e;t)$ constant *mont* asserts that its argument denotes a Montagovian (i.e. non-abstract) individual. The introduction of this constant is required by the inclusion of 'propositional' individuals in the domain of the TY_2^3 type e (cf. (7.2.16)), and by the use of (7.4.1) to identify the WTY_1^3 cor-

[8]The only exceptions include simple existential sentences, as we will see below.

[9]This possibility has been pointed out by Jeroen Groenendijk.

relates of *Montagovian* individuals:

(7.4.1) $\lambda p.[\exists x.\, mont(x) \wedge p = (\lambda i.E(x,i))]$

(7.4.2) $\lambda p.p \wedge [\neg\exists x.\, mont(x) \wedge p = (\lambda i.E(x,i))]$

The second conjunct of the predicate from (7.4.2) is required by a version of Montague's homomorphism requirement on the syntax-semantics map, cf. (**Montague, 1970b**). This requirement demands that no two occurrences of a logical form may receive an interpretation in different (sub-)types (cf. Obs. 1.1.ii). For convenience, we will hereafter abbreviate the predicates from (7.4.1) and (7.4.2) by '$\imath$' and 'ρ', respectively.

From $(s;t)_\imath$ and $(s;t)_\rho$, the subtypes of all other WTY_1^3 types are obtained via a sorted variant of the type-forming rule from Definition 7.1.1:

Definition 7.4.1 (WTY_1^3 sorts). The set $\mathsf{w1Sorts}$ of WTY_1^3 *subtypes* (or *sorts*) is the smallest set of strings s.t., for $0 \leq n \in \mathbb{N}$, if $\alpha_1, \ldots, \alpha_n \in \mathsf{w1Sort}$, then $(\alpha_1 \ldots \alpha_n\, s;t_\imath) \in \mathsf{w1Sort}$ and $(\alpha_1 \ldots \alpha_n\, s;t_\rho) \in \mathsf{w1Sort}$.

The above identifies the subtype $((s;t)_\imath\, s;t_\rho)$ as the sort for functions from objects of the sort $(s;t)_\imath$ to objects of the sort $(s;t)_\rho$. By the definition of the basic WTY_1^3 sorts $(s;t)_\imath$ and $(s;t)_\rho$, these functions are associated with the WTY_1^3 representations of properties of individuals. The semantic constraints from Definition 7.2.2 restrict the sort $((s;t)_\imath\, s;t_\rho)$ to sets of objects satisfying some predicate of the form $\lambda\boldsymbol{x}\lambda i.P\big([\imath x.\boldsymbol{x} = (\lambda j.E(x,j))], i\big)$. As a result, the replacement of WTY_1^3 types by *sorts* reduces single-type domains to sets of 'representationally relevant' objects (in the sense described in Ch. 4.1.2).

The replacement of WTY_1^3 types by sorts in Tables 7.1 and 7.2 (cf. Tables 7.5, resp. 7.6, next page) restricts the translations of logical PTQ forms to WTY_1^3 terms of the specified sort. In particular, the restriction of the WTY_1^3 constant $\boldsymbol{E}$ to arguments of the sort $(s;t)_\imath$ prevents the application of $\boldsymbol{E}$ to the sort-$(s;t)_\rho$ term $\boldsymbol{walk}(\boldsymbol{bill})$. This fact accounts for the syntactic ill-formedness of the expression Bill walks exists from (18c).

The possibility of providing a semantic basis for *all* syntactic well-formedness constraints is curbed by the assignment of the sort $(s;t)_\imath$ to simple existential sentences.[10] As a result, the presented variant of our WTY_1^3-based single-type semantics will be unable to explain the well-formedness of the logical form from (23), and the ill-formedness of the expressions from (24a) and (24b) (next page; cf. (18c), (18b)). In the single-type semantics from Section 7.2, the last two expressions are associated with the designators of the proposition 'Bill walks'.

[10] See the definition of the WTY_1^3 translation of the sentence John does not exist from (7.2.33).

CONSTANT	WTY_1^3 SORT
$\bigwedge, \bigvee$	$(\alpha\,(s;t)\,s;t_\rho)$
$\dot{\Rightarrow}, \dot{=}, \dot{\neq}, \dot{\rightarrow}, \dot{\leftrightarrow}$	$(\alpha\,\alpha\,s;t_\rho)$
$\top, \bot, \boldsymbol{w}$	$(s;t)_\rho$
john, ***mary***, ***bill***, ***ninety***, ***sherlock***, ***pat***, ***moriarty***, ***partee***	$(s;t)_,$
$\boxdot, \diamond$	$((s;t)_\rho\,s;t_\rho)$
man, ***woman***, ***park***, ***fish***, ***pen***, ***unicorn***, ***room***, ***problem***	$((s;t)_,\,s;t_\rho)$
run, ***walk***, ***talk***, ***wait***, ***arrive***, ***E***	$((s;t)_,\,s;t_\rho)$
find, ***lose***, ***eat***, ***love***, ***date***, ***remember***, ***fear***, ***destroy***, ***hate***, ***enter***	$((s;t)_,\,(s;t)_,\,s;t_\rho)$
believe, ***assert***	$((s;t)_\rho\,(s;t)_,\,s;t_\rho)$
temp, ***price***, ***rise***, ***change***	$((s;t)_\rho\,(s;t)_,\,s;t_\rho)$
rapidly, ***slowly***, ***voluntarily***, ***allegedly***	$(((s;t)_,\,s;t_\rho)\,(s;t)_,\,s;t_\rho)$
try, ***wish***	$(((s;t)_,\,s;t_\rho)\,(s;t)_,\,s;t_\rho)$
in, ***for***	$((s;t)_,\,((s;t)_,\,s;t_\rho)\,(s;t)_,\,s;t_\rho)$
seek, ***conceive***	$((((s;t)_,\,s;t_\rho)\,s;t_\rho)\,(s;t)_,\,s;t_\rho)$
about	$((((s;t)_,\,s;t_\rho)\,s;t_\rho)\,((s;t)_,\,s;t_\rho)\,(s;t)_,\,s;t_\rho)$

TABLE 7.5. Sorted $\mathcal{L}^{w1}$ constants.

VARIABLE	WTY_1^3 SORT
$\boldsymbol{x}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n, \boldsymbol{y}, \boldsymbol{z}$	$(s;t)_,$
$\boldsymbol{p}, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_n, \boldsymbol{q}, \boldsymbol{r}$	$(s;t)_\rho$
$\boldsymbol{P}, \boldsymbol{P}_1, \ldots, \boldsymbol{P}_n$	$((s;t)_,\,s;t_\rho)$
$\boldsymbol{Q}, \boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_n$	$(((s;t)_,\,s;t_\rho)\,s;t_\rho)$
$\boldsymbol{L}, \boldsymbol{L}_1, \ldots, \boldsymbol{L}_n$	$((((s;t)_,\,s;t_\rho)\,s;t_\rho)\,(s;t)_,\,s;t_\rho)$

TABLE 7.6. Sorted WTY_1^3 variables.

(23) Possibly [$_{\mathrm{S}}$Bill exists].

(24) a. * [$_{\mathrm{S}}$Bill exists] walks.

b. * [$_{\mathrm{CP}}$That Bill exists] walks.

Notably, expressions of the above form only constitute a very small percentage of PTQ expressions. Since simple existential sentences are further only rarely used in ordinary (i.e. non-philosophical) discourse, the inability of our sorted single-type semantics to explain the well- (or ill-)formedness of the expressions from (23)

(resp. (24a), (24b)) is rather unproblematic. In Figure 7.2, the residual mismatch[11] between categories and sorts is represented by stacked diagonals, ⫽.

To regain the full explanatory power of the Montagovian type system in our single-type semantics (cf. Fig. 7.3, below), one could introduce a second existence predicate into the logic TY_2^3 (next to the constant E from Ch. 6.1).[12] A possible candidate for this predicate would be Linsky and Zalta's concreteness predicate $C!$, cf. (**Linsky and Zalta, 1994**). The replacement of the existence predicate E by the predicate $C!$ in the definition of the WTY_1^3 constant $\boldsymbol{E}$ from (C5) (in (7.4.3)) would then prevent the identification of simple existential sentences (or CPs) as sort-$(s;t)_\iota$ terms. The resulting semantics would recognize all Montagovian categorial distinctions, and would, thus, explain the well- (resp. ill-) formedness of the logical forms from (23) (resp. (24a), (24b)).

$$\boldsymbol{E} \;=\; \lambda\boldsymbol{x}\lambda i.C!\left([\iota x.\boldsymbol{x}=(\lambda j.E(x,j))],i\right) \tag{7.4.3}$$

An answer to the question of whether the introduction of a second non-logical existence predicate is well-justified is left to the reader. We will see in Chapter 8.3 that the different subtypes of the 'strong' single-type type $(s\,s;t)$ restore Montague's semantic distinctions without the metatheoretical introduction of a second existence predicate.

(GB) Syntax	S	NP	N	C	SAV	⋯
WTY_1^3 Semantics (Object theory)	$(s;t)_\rho$	$(s;t)_\iota$	$((s;t)_\iota, s;t_\rho)$	$((s;t)_\rho\, s;t_\rho)$		⋯

FIGURE 7.3. Syntactic categories and WTY_1^3 subtypes (alt).

This completes our discussion of the explanatory power of our weak single-type semantics. We close the chapter with a summary of our interim achievements.

7.5. Summary

This chapter has developed a weak single-type semantics for the set of English logical forms from (**Montague, 1973**). This semantics is a designated model for a subsystem, WTY_1^3, of the logic TY_2^3, which interprets the forms' TY_0 translations from Part I into constructions out of propositions. In virtue of the possibility of representing individuals in the propositional type $(s;t)$, proper names, sentences,

[11]The reader is reminded that the same-type interpretation of complementizers and sentence adverbs is a Montagovian heritage (cf. Ch. 3.4).

[12]I owe this observation to Lucas Champollion.

and complement phrases receive an interpretation in this type (cf. Prop. 1.2). The resulting semantics preserves the merits of the 'pure' single-type semantics from Part I. It improves upon this semantics by explaining the truth-aptness of proper names (cf. Prop. 1.3.i) and by identifying their equivalent simple existential sentences.

In particular, the assignment of truth-conditions to names (Prop. 1.3.i) is warranted by the interpretation of names as functions from situations to truth-combinations. The identification of a name's sentential equivalents is enabled by the interpretation of names into the set of situations in which their traditional type-e referents exist. Since we have defined the existence predicate E as a bivalent predicate, proper names will receive a classical truth-value at every situation. Since only the WTY^3_1 translations of simple *existential* sentences are defined by TY^3_2 formulas of the described form, the set of a name's WTY^3_1-equivalent sentences is restricted to existential sentences. This observation motivates our description of the semantics of the type $(s;t)$ as a *weak* single-type semantics (cf. Ch. 5.2).

The next chapter is dedicated to the development of a *strong* single-type semantics, which supports the claim from Proposition 1.3.ii.

CHAPTER 8

Strong Single-Type Semantics

The present chapter provides a *strong* single-type semantics for Montague's PTQ fragment in the sense of Chapter 5.2.1. This semantics is a designated model for a subsystem of the logic TY_2^3 which interprets names, sentences, and complement phrases in the type for *propositional concepts*, $(s;(s;t))$ (or, equivalently, in the type for characteristic functions of binary relations between situations, $(s\,s;t)$). Objects of this type represent individuals via functions from situations w to the set of situations at which the designators of all w-true propositions about the individuals are true, and represent propositions via functions from situations w to the set of situations at which the propositions' designators and all w-true propositions about their type-e arguments are true. The semantics improves upon the semantics from the previous chapter by identifying a name's value at a situation with the values of contextually contingent sentences at that situation (cf. Prop. 1.3.ii).

Our strategy for the presentation of a strong single-type semantics follows the strategy for the presentation of its weak counterpart from Chapter 7. Thus, to obtain a strong $(s\,s;t)$-based semantics for the PTQ fragment, we will first define the $(s\,s;t)$-based subsystem, STY_1^3, of the logic TY_2^3 (in Sect. 8.1) and introduce its designated language, frame, and interpretation function (in Sect. 8.2.1). We will then adapt the WTY_1^3 translations of basic PTQ words to terms of the logic STY_1^3, and formulate a number of constraints on the interpretation of the adapted translations (in Sect. 8.2.2). Our constraints on these translations will differ in interesting respects from the constraints for terms of the logic WTY_1^3. Section 8.3 gives the truth- and equivalence-conditions for logical PTQ forms of the basic STY_1^3 type. These conditions contain a solution to the problem of explaining syntactic well-formedness in single-type semantics. The chapter closes with a discussion of the representational complexity of strong single-type semantics, and with a comparison of this semantics to classical propositional logics with only the Sheffer stroke.

8.1. The 'Strong' Object Theory STY_1^3

We again begin by defining the object theory, STY_1^3, of the presented single-type semantics. This theory is a subsystem of the logic TY_2^3 which constructs all its types from the TY_2^3 type for propositional concepts, $(s\,s;t)$. Like the basic type of

the logic WTY_1^3, the type $(s\,s;t)$ is a propositional conjoinable TY_2^3 type, whose domain has an algebraic structure. Since strong single-type models are partial models (cf. the superscript '3'), objects in these models further satisfy the requirement on suitable single basic types from Chapter 4.2.4.

The name of the logic STY_1^3 shares the rationale behind the name of the $(s;t)$-based logic WTY_1^3. To emphasize the possibility of providing semantically 'rich' representations of individuals and propositions in models of this logic (cf. Ch. 5.2.1), we replace the letter 'W' (for '*weak*') by 'S' (for '*strong*').

8.1.1. Types and Terms. Our survey of Montagovian objects from Chapter 4.2.3 has identified characteristic functions of binary relations between situations (or *propositional concepts*) as the simplest strong single-type candidate. As a result, we adopt the TY_2^3 type $(s\,s;t)$ as the lowest-rank type of the logic STY_1^3. From this type, the set of STY_1^3 types is defined in analogy to the set of WTY_1^3 types as follows:

Definition 8.1.1 (STY_1^3 types). The set $\mathsf{s1Type}$ of STY_1^3 *types* is the smallest set of strings s.t., if $\alpha_1, \ldots, \alpha_n \in \mathsf{s1Type}$, then $(\alpha_1 \ldots \alpha_n\, s\, s; t) \in \mathsf{s1Type}$.

The class of languages for the logic STY_1^3 is a type-$(s\,s;t)$ variant of the class of languages for the logic WTY_1^3. In particular, a language L^{s1} and a set of variables $\mathcal{V}^{s1}$ for the logic STY_1^3 are proper subsets, $L^2_{\restriction\mathsf{s1Type}}$, resp. $\mathcal{V}^2_{\restriction\mathsf{s1Type}}$, of the relevant sets of non-logical constants and variables of the logic TY_2^3. From their members, complex STY_1^3 terms are formed inductively through suitably typed[1] variants of the clauses from Definition 7.1.2 and through the non-logical constant $\vdash$ (called *focused negation*):

Definition 8.1.2 (STY_1^3 terms). Let $\alpha_1, \ldots, \alpha_n, \beta \in \mathsf{s1Type}$. The set T^{s1}_α of STY_1^3 *terms* of the type α is then defined as follows:

(i) $L^{s1}_\alpha, \mathcal{V}^{s1}_\alpha \subseteq T^{s1}_\alpha$, $\textcircled{\scriptsize\perp} \in T^{s1}_{(s\,s;t)}$;
(ii) If $\boldsymbol{B} \in T^{s1}_{(\beta\alpha_1\ldots\alpha_n\,s\,s;t)}$ and $\boldsymbol{A} \in T^{s1}_\beta$, then $(\boldsymbol{B}(\boldsymbol{A})) \in T^{s1}_{(\alpha_1\ldots\alpha_n\,s\,s;t)}$;
(iii) If $\boldsymbol{A} \in T^{s1}_{(\alpha_1\ldots\alpha_n\,s\,s;t)}$ and $\boldsymbol{x} \in \mathcal{V}_\beta$, then $(\lambda\boldsymbol{x}.\boldsymbol{A}) \in T^{s1}_{(\beta\alpha_1\ldots\alpha_n\,s\,s;t)}$;
(iv) If $\boldsymbol{B}, \boldsymbol{C} \in T^{s1}_\alpha$, then $(\boldsymbol{B} \doteRightarrow \boldsymbol{C}) \in T^{s1}_{(s\,s;t)}$;
(v) If $\boldsymbol{B} \in T^{s1}_\alpha$, then $\vdash\boldsymbol{B} \in T^{s1}_\alpha$.

Our introduction of focused negation (clause (v)) is motivated by the unsuitability of the TY_2^3 symbol $\neg$ for the STY_1^3 translation of English negation, as we will see in Section 8.2.2.

From the non-logical STY_1^3 constants $\textcircled{\scriptsize\perp}$ and $\doteRightarrow$, suitably typed variants of the designated constants from Notation 2.1.1 are, again, easily obtained. In particu-

[1] As a result, $\textcircled{\scriptsize\perp}$ and $\doteRightarrow$ are non-logical STY_1^3 constants of the type $(s\,s;t)$, resp. $(\alpha\,\alpha\,s\,s;t)$.

lar, the constants $\textcircled{\scriptsize T}$; $\boxdot$ and $\diamondsuit$; $\bigwedge$ and $\bigvee$; and $\doteq$, $\dot{\rightarrow}$, and $\dot{\leftrightarrow}$ are now characterized as non-logical STY^3_1 constants of the types $(s\,s,t)$, $((s\,s;t)\,s\,s;t)$, $(\alpha\,(s\,s;t)\,s\,s;t)$, and $(\alpha\,\alpha\,s\,s;t)$, respectively, where $\alpha \in \mathsf{s1Type}$. The TY^3_2 definition of these constants (in Sect. 8.2.2) will ensure their desired semantic behavior.

8.1.2. Models. Like general frames for the logic WTY^3_1, general STY^3_1 frames are restricted variants of general frames for the logic TY^3_2:

Definition 8.1.3 (General STY^3_1 frames). A *general* STY^3_1 *frame* F^{s1} is a proper subset, $F^2_{\restriction \mathsf{s1Type}} = \{D^{F^2}_\alpha \mid \alpha \in \mathsf{s1Type}\}$, of the relevant TY^3_2 frame.

Since we have identified the TY^3_2 domain $D_{(s\,s;t)}$ with a subset of the function space $((W \times W) \rightarrow \mathbf{3})$, the ground domain of the logic STY^3_1 will contain the desired partial objects from Chapter 4.2.4.

STY^3_1 interpretation functions $I_{F^{s1}}$ and assignments $g_{F^{s1}}$ are defined as subsets of the relevant functions in the logic TY^3_2 (in (8.1.1), resp. (8.1.2)). General models for STY^3_1 constitute subsets of general De Morgan models for the logic TY^3_2 (in (8.1.3)):

$$I_{F^2 \restriction \mathsf{s1Type}} : L^2_{\restriction \mathsf{s1Type}} \rightarrow F^2_{\restriction \mathsf{s1Type}} \tag{8.1.1}$$

$$g_{F^2 \restriction \mathsf{s1Type}} : \mathcal{V}^2_{\restriction \mathsf{s1Type}} \rightarrow F^2_{\restriction \mathsf{s1Type}} \tag{8.1.2}$$

$$M_{F^2 \restriction \mathsf{s1Type}} = \langle F^2_{\restriction \mathsf{s1Type}}, I_{F^2 \restriction \mathsf{s1Type}}, V_{F^2 \restriction \mathsf{w1Type}} \rangle \tag{8.1.3}$$

This completes our discussion of the interpretation of STY^3_1 terms. We next turn to the notions of STY^3_1 truth, equivalence, and entailment.

8.1.3. Truth and entailment. Since the propositional type $(s;t)$ is not available in the logic STY^3_1, basic-type terms of this logic do not admit of a TY^3_2 truth-definition. However, we will see in Section 8.3 that truth- and falsity-conditions for the STY^3_1 translations of names and sentences can be obtained in a straightforward manner.

Like entailment in the logic WTY^3_1, the STY^3_1 entailment relation is a restricted variant of generalized entailment for the logic TY^3_2:

Definition 8.1.4 (STY^3_1 entailment). A set of basic-type STY^3_1 terms $\Gamma = \{\gamma \mid \gamma \in T^{s1}_{(s\,s;t)}\}$ *entails* a set of STY^3_1 terms $\Delta = \{\delta \mid \delta \in T^{s1}_{(s\,s;t)}\}$, i.e. $\Gamma \models_g \Delta$, if, for all general WTY^3_1 models $M_{F^{s1}}$ and assignments $g_{F^{s1}}$,

$$\bigcap_{\gamma \in \Gamma} V_{F^{s1}}(g_{F^{s1}}, \gamma) \subseteq \bigcup_{\delta \in \Delta} V_{F^{s1}}(g_{F^{s1}}, \delta)\,.$$

The relation of STY^3_1 equivalence is then a type-$(s\,s;t)$ variant of the equivalence relation between terms of the logic WTY^3_1. The proof theory of the logic STY^3_1 has the expected form.

This completes our discussion of the $(s\,s;t)$-based single-type logic STY_1^3. We next show that this logic enables the 'strong' single-type interpretation of proper names, CPs, and sentences along the lines described in Chapter 4.2.3.

8.2. An STY_1^3 Semantics for the PTQ Fragment

Following the structure of the presentation from Chapter 7.2, we first specify the designated STY_1^3 language $\mathcal{L}^{s1}$ and frame $\mathcal{F}^{s1}$. The latter are the particular STY_1^3 language, respectively frame whose elements are associated with the designated STY_1^3 constants from Section 8.1.1 and with the lexical elements from Table 3.1.

8.2.1. Fixing $\mathcal{L}^{s1}$ and $\mathcal{F}^{s1}$. The members of the STY_1^3 language $\mathcal{L}^{s1}$ and the set of STY_1^3 variables $\mathcal{V}^{s1}$ (in Tables 8.1, resp. 8.2) are STY_1^3-typed variants of the non-logical constants, resp. variables of the WTY_1^3 language $\mathcal{L}^{w1}$ and the set of WTY_1^3 variables $\mathcal{V}^{w1}$ from Tables 7.1 and 7.2. To enable the adequate translation of linguistic negation in our strong single-type semantics, the language $\mathcal{L}^{s1}$ contains the symbol for focused negation, $\sqsubset$.

Constant	STY_1^3 Type
$\boldsymbol{B}$, $\boldsymbol{C}$	$(\alpha_1 \ldots \alpha_n\, s\, s;t)$
$\sqsubset$, $\neg$	$((\alpha_1 \ldots \alpha_n\, s\, s;t)\, \alpha_1 \ldots \alpha_n\, s\, s;t)$
$\wedge$, $\vee$	$((\alpha_1 \ldots \alpha_n\, s\, s;t)\, (\alpha_1 \ldots \alpha_n\, s\, s;t)\, \alpha_1 \ldots \alpha_n\, s\, s;t)$
$\bigwedge$, $\bigvee$	$(\alpha\, (s\, s;t)\, s\, s;t)$
$\dot{\Rightarrow}$, $\dot{=}$, $\dot{\neq}$, $\dot{\rightarrow}$, $\dot{\leftrightarrow}$	$(\alpha\, \alpha\, s\, s;t)$
$\top$, $\bot$, ***john***, ***mary***, ***bill***, ***ninety***, ***sherlock***, ***moriarty***, ***pat***, ***partee***, ***w***	$(s\, s;t)$
$\Box$, $\Diamond$, ***man***, ***woman***, ***park***, ***fish***, ***pen***, ***room***, ***unicorn***, ***problem***	$((s\, s;t)\, s\, s;t)$
run, ***walk***, ***talk***, ***wait***, ***arrive***, ***E***	$((s\, s;t)\, s\, s;t)$
find, ***lose***, ***eat***, ***love***, ***date***, ***remember***, ***fear***, ***destroy***, ***hate***, ***enter***, ***believe***, ***assert***	$((s\, s;t)\, (s\, s;t)\, s\, s;t)$
temp, ***price***, ***rise***, ***change***	$(((s\, s;t)\, s\, s;t)\, s\, s;t)$
rapidly, ***slowly***, ***voluntary***, ***allegedly***	$(((s\, s;t)\, s\, s;t)\, (s\, s;t)\, s\, s;t)$
try, ***wish***	$(((s\, s;t)\, s\, s;t)\, (s\, s;t)\, s\, s;t)$
in, ***for***	$((s\, s;t)\, ((s\, s;t)\, s\, s;t)\, (s\, s;t)\, s\, s;t)$
seek, ***conceive***	$((((s\, s;t)\, s\, s;t)\, s\, s;t)\, (s\, s;t)\, s\, s;t)$
about	$((((s\, s;t)\, s\, s;t)\, s\, s;t)\, ((s\, s;t)\, s\, s;t)\, (s\, s;t)\, s\, s;t)$

Table 8.1. $\mathcal{L}^{s1}$ constants.

VARIABLE	STY_1^3 TYPE
$\boldsymbol{x}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{p}, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_n, \boldsymbol{q}, \boldsymbol{r}$	$(s\, s; t)$
$\boldsymbol{P}, \boldsymbol{P}_1, \ldots, \boldsymbol{P}_n, \boldsymbol{T}, \boldsymbol{T}_1, \ldots, \boldsymbol{T}_n$	$((s\, s; t)\, s\, s; t)$
$\boldsymbol{Q}, \boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_n$	$(((s\, s; t)\, s\, s; t)\, s\, s; t)$
$\boldsymbol{L}, \boldsymbol{L}_1, \ldots, \boldsymbol{L}_n$	$((((s\, s; t)\, s\, s; t)\, s\, s; t)\, (s\, s; t)\, s\, s; t)$

TABLE 8.2. STY_1^3 variables.

Again, we let the STY_1^3 frame $\mathcal{F}^{s1}$ be very large, and assume that the designated STY_1^3 interpretation function $\mathcal{I}_{\mathcal{F}^{s1}} : \mathcal{L}^{s1} \to \mathcal{F}^{s1}$ respects the conventional lexical relations between content words. To allow the definition of STY_1^3 terms through terms of the logic TY_2^3, we expect that the designated TY_2^3 language $\mathcal{L}^2$ and set of variables $\mathcal{V}^2$ (cf. Tables 7.3, 7.4) also include the STY_1^3 language $\mathcal{L}^{s1}$ and set $\mathcal{V}^{s1}$. Further, we require that the frame $\mathcal{F}^2$ and interpretation function $\mathcal{I}_{\mathcal{F}^2}$ of the designated model for the logic TY_2^3 embed the designated STY_1^3 frame and interpretation function, such that $\mathcal{F}^{s1} = \mathcal{F}^{2\restriction \mathsf{s1Type}}$ and $\mathcal{I}_{\mathcal{F}^{s1}} = \mathcal{I}_{\mathcal{F}^{2}\restriction \mathsf{s1Type}}$.

8.2.2. Translations of Logical PTQ Forms. The STY_1^3 translations of the lexical elements from Table 3.1 are STY_1^3-typed variants of the elements' translations from Definition 7.2.1. Thus, the STY_1^3 translation, ***bill***, of the name Bill is the type-$(\underline{s}\, s; t)$ term ***bill*** from Table 8.1. The STY_1^3 translation, ***walk***, of the intransitive verb walk has the type $((\underline{s}\, s; t)\, \underline{s}\, s; t)$. The STY_1^3 translations of non-lexical forms are $(s\, s; t)$-based variants of the translations from (3.2.1) to (3.2.27).

Since basic-type terms of the logic STY_1^3 have a different type from their corresponding terms in the logic WTY_1^3 – and since they encode 'richer' semantic information than their WTY_1^3 correspondents (cf. Ch. 4.2.3, 5.2.1) –, the interpretations of basic-type STY_1^3 terms are governed by a different set of semantic constraints than the ones from Definition 7.2.2. For the interpretation of the designated STY_1^3 constants, these constraints are given in Definition 8.2.2 (next page). In the constraints, the function $^\circ$ is the inverse image of the meaning-shifting function $^\bullet$ from Definition 8.2.1. We will hereafter write the symbol '$^\bullet$' in postfix notation, such that '$x^\bullet$' denotes $^\bullet(x)$. For variables of the logic TY_2^3, we use the typing conventions from Table 7.4.

DEFINITION 8.2.1. The function $^\bullet$ sends TY_2^3 terms to constructions out of the type $(s\, s; t)$, such that

$$
\begin{array}{lllll}
\text{(i)} & x^\bullet & = & (\lambda i \lambda j \forall p.(p(i) \wedge abt(x,p)) \twoheadrightarrow p(j)), \\
 & q^\bullet & = & (\lambda i \lambda j.q(j) \wedge (\forall p.(p(i) \wedge (\exists x.abt(x,q) \wedge abt(x,p))) \twoheadrightarrow p(j))); \\
\text{(ii)} & (B(A))^\bullet & = & (B^\bullet(A^\bullet)); \\
\text{(iii)} & (\lambda x_\alpha \lambda i.A(i))^\bullet & = & (\lambda x_\alpha^\bullet \lambda i.A^\bullet(i)).
\end{array}
$$

Our use of the function $^\bullet$ suggests that all 'relevant' STY_1^3 objects (in the sense of Ch. 4.1.2) are constructions out of type-$(s\,s;t)$ representations of individuals and propositions (cf. (4.2.18), resp. (4.2.19)). In particular, clause (i) describes the shifting of individual-denoting terms x to the designator of a function from ordered pairs of situations $\langle w_1, w\rangle$ to the truth-value of the proposition 'the designators of all true propositions at w_1 which carry information about x are true at w'. Further, it describes the lifting of intensional formulas q to the designator of a function from pairs of situations $\langle w_1, w\rangle$ to the truth-value of the proposition 'q is true at w and the designators of all true propositions at w_1 which carry information about one of q's aboutness subjects are also true at w'.

Clauses (ii) and (iii) generalize the definition of the function $^\bullet$ to TY_2^3 terms of arbitrarily complex types. For example, the function $^\bullet$ lifts the type-$(e\,s;t)$ term P to its correspondent in the type $((s\,s;t)\,s\,s;t)$ as follows:

(8.2.1)

$$
\begin{aligned}
& P^\bullet \\
= \;& (\lambda x \lambda i.P(x,i))^\bullet && \text{(by } \eta\text{-expansion)} \\
= \;& \lambda x^\bullet \lambda i.[\lambda j.P(x,j)]^\bullet(i) && \text{(by (iii))} \\
= \;& \lambda x^\bullet \lambda i.\big[\lambda k \lambda j.P(x,j) \wedge && \text{(by (i), case } q^\bullet\text{)} \\
& \qquad \big(\forall p.(p(k) \wedge (\exists y.\, abt(y,[\lambda k_1.P(x,k_1)]) \wedge abt(y,p))) \twoheadrightarrow p(j)\big)\big](i) \\
= \;& \lambda x^\bullet \lambda i \lambda j.P(x,j) \wedge \big(\forall p.(p(i) \wedge (\exists y.\, abt(y,[\lambda k.P(x,k)]) \wedge abt(y,p))) \twoheadrightarrow p(j)\big)
\end{aligned}
$$

The constraints on the interpretation of the members of $\mathcal{L}^{s1}$ are specified below. To avoid specifying the defining TY_2^3 term for every member of $\mathcal{L}^{s1}$, we only define a few representative constants for each STY_1^3 type. We assume that $\boldsymbol{c}$, π, $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{P}$, $\boldsymbol{T}$, and $\boldsymbol{Q}$ are defined by the TY_2^3 terms $c^\bullet$, $\varphi^\bullet$, $x^\bullet$, $y^\bullet$, $P^\bullet$, $T^\bullet$, and $Q^\bullet$, respectively. The function $^\circ$ is defined as $(^\bullet)^{-1}$.

Definition 8.2.2 (Definition of $\mathcal{L}^{s1}$ constants). The interpretations of the STY_1^3 constants from Table 8.1 obey the following constraints:

$$
\begin{aligned}
&\text{(C1)} && \textcircled{\perp} && = && \lambda i \lambda j.\perp; \\
&\text{(C2)} && (\boldsymbol{B} \dot{\Rightarrow} \boldsymbol{C}) && = && \lambda i \lambda j.\boldsymbol{B}(i,j) \Rightarrow \boldsymbol{C}(i,j); \\
&\text{(C14)} && \urcorner\pi && = && (\neg\varphi)^\bullet; \\
&\text{(C0)} && \boldsymbol{w} && = && \big(\lambda i \forall p.p(@) \twoheadrightarrow p(i)\big)^\bullet; \\
&\text{(C3)} && \boldsymbol{bill} && = && \lambda i \lambda j \forall p.(p(i) \wedge abt(bill,p)) \twoheadrightarrow p(j); \\
&\text{(C4)} && \boldsymbol{man} && = && \lambda \boldsymbol{x} \lambda i \lambda j.man(\boldsymbol{x}^\circ, j) \wedge \big(\forall p.(p(i) \wedge \\
& && && && \qquad (\exists y.\, abt(y,[\lambda k.man(\boldsymbol{x}^\circ,k)]) \wedge abt(y,p))) \twoheadrightarrow p(j)\big); \\
&\text{(C5)} && \boldsymbol{walk} && = && \lambda \boldsymbol{x} \lambda i \lambda j.walk(\boldsymbol{x}^\circ, j) \wedge \big(\forall p.(p(i) \wedge \\
& && && && \qquad (\exists y.\, abt(y,[\lambda k.walk(\boldsymbol{x}^\circ,k)]) \wedge abt(y,p))) \twoheadrightarrow p(j)\big);
\end{aligned}
$$

$$\text{(C6)}\ \ \textbf{\textit{believe}} = \lambda\boldsymbol{p}\,\lambda\boldsymbol{x}\,\lambda i\lambda j.\,\mathit{believe}(\boldsymbol{p}^\circ,\boldsymbol{x}^\circ,j)\wedge\big(\forall p.(p(i)\wedge(\exists y.\,\mathit{abt}(y,p)\wedge \mathit{abt}(y,[\lambda k.\,\mathit{believe}(\boldsymbol{p}^\circ,\boldsymbol{x}^\circ,k)]))) \twoheadrightarrow p(j)\big);$$

$$\text{(C7)}\ \ \textbf{\textit{find}} = \lambda\boldsymbol{y}\,\lambda\boldsymbol{x}\,\lambda i\lambda j.\,\mathit{find}(\boldsymbol{y}^\circ,\boldsymbol{x}^\circ,j)\wedge\big(\forall p.\big(p(i)\wedge\big(\exists z.\,\mathit{abt}(z,p)\wedge \mathit{abt}\big(z,[\lambda k.\,\mathit{find}(\boldsymbol{y}^\circ,\boldsymbol{x}^\circ,k)]\big)\big)\big) \twoheadrightarrow p(j)\big);$$

$$\text{(C8)}\ \ \textbf{\textit{temp}} = \lambda\boldsymbol{T}\,\lambda i\lambda j.\,\mathit{temp}(\boldsymbol{T}^\circ,j)\wedge\big(\forall p.\big(p(i)\wedge\big(\exists z.\,\mathit{abt}(z,p)\wedge \mathit{abt}\big(z,[\lambda k.\,\mathit{temp}(\boldsymbol{T}^\circ,k)]\big)\big)\big) \twoheadrightarrow p(j)\big);$$

$$\text{(C9)}\ \ \textbf{\textit{rapidly}} = \lambda\boldsymbol{P}\,\lambda\boldsymbol{x}\,\lambda i\lambda j.\,\mathit{rapidly}(\boldsymbol{P}^\circ,\boldsymbol{x}^\circ,j)\wedge\big(\forall p.\big(p(i)\wedge\big(\exists y.\,\mathit{abt}(y,p)\wedge \mathit{abt}\big(y,[\lambda k.\,\mathit{rapidly}(\boldsymbol{P}^\circ,\boldsymbol{x}^\circ,k)]\big)\big)\big) \twoheadrightarrow p(j)\big);$$

$$\text{(C10)}\ \ \textbf{\textit{try}} = \lambda\boldsymbol{P}\,\lambda\boldsymbol{x}\,\lambda i\lambda j.\,\mathit{try}(\boldsymbol{P}^\circ,\boldsymbol{x}^\circ,j)\wedge\big(\forall p.\big(p(i)\wedge\big(\exists y.\,\mathit{abt}(y,p)\wedge \mathit{abt}\big(y,[\lambda k.\,\mathit{try}(\boldsymbol{P}^\circ,\boldsymbol{x}^\circ,k)]\big)\big)\big) \twoheadrightarrow p(j)\big);$$

$$\text{(C11)}\ \ \textbf{\textit{in}} = \lambda\boldsymbol{y}\,\lambda\boldsymbol{P}\,\lambda\boldsymbol{x}\,\lambda i\lambda j.\,\mathit{in}(\boldsymbol{y}^\circ,\boldsymbol{P}^\circ,\boldsymbol{x}^\circ,j)\wedge\big(\forall p.\big(p(i)\wedge\big(\exists z.\,\mathit{abt}(z,p)\wedge \mathit{abt}\big(z,[\lambda k.\,\mathit{in}(\boldsymbol{y}^\circ,\boldsymbol{P}^\circ,\boldsymbol{x}^\circ,k)]\big)\big)\big) \twoheadrightarrow p(j)\big);$$

$$\text{(C12)}\ \ \textbf{\textit{seek}} = \lambda\boldsymbol{Q}\,\lambda\boldsymbol{x}\,\lambda i\lambda j.\,\mathit{seek}(\boldsymbol{Q}^\circ,\boldsymbol{x}^\circ,j)\wedge\big(\forall p.\big(p(i)\wedge\big(\exists z.\,\mathit{abt}(z,p)\wedge \mathit{abt}\big(z,[\lambda k.\,\mathit{seek}(\boldsymbol{Q}^\circ,\boldsymbol{x}^\circ,k)]\big)\big)\big) \twoheadrightarrow p(j)\big);$$

$$\text{(C13)}\ \ \textbf{\textit{about}} = \lambda\boldsymbol{Q}\,\lambda\boldsymbol{P}\,\lambda\boldsymbol{x}\,\lambda i\lambda j.\,\mathit{about}(\boldsymbol{Q}^\circ,\boldsymbol{P}^\circ,\boldsymbol{x}^\circ,j)\wedge\big(\forall p.\big(p(i)\wedge\big(\exists z.\ \mathit{abt}(z,p)\wedge \mathit{abt}\big(z,[\lambda k.\,\mathit{about}(\boldsymbol{Q}^\circ,\boldsymbol{P}^\circ,\boldsymbol{x}^\circ,k)]\big)\big)\big) \twoheadrightarrow p(j)\big)$$

The constraints (C1) and (C2) define the designated STY_1^3 constants $\textcircled{\scriptsize\perp}$ and $\dot{\Rightarrow}$ as the results of lifting the TY_2^3 connectives $\perp$ and $\Rightarrow$ to constructions out of the basic STY_1^3 type $(s\,s;t)$.

The constraint (C14) defines the focused negation of basic-type STY_1^3 terms as the result of negating the 'updating' proposition[2] in the definition of these terms through the use of the familiar negation operator $\neg$. The definition of STY_1^3 terms of the form $\ulcorner(\varphi^\bullet)$ employs this strategy. Since the type-$(s\,s;t)$ representation of individuals does not express an update on the available information at the relevant situation[3] – and since linguistic negation is typically not available for proper names –, we leave the negation of STY_1^3 representations of individuals undefined. From STY_1^3 terms of the basic type, the definition of $\ulcorner$ can be lifted to suitably defined terms of any complex STY_1^3 type.

From (C1), (C2), and Notation 2.1.1, the STY_1^3 proxies for all other TY_2^3 connectives are again easily defined:

NOTATION 8.2.1. We write $\quad \neg\boldsymbol{B}$ for $(\lambda\boldsymbol{x}\lambda i\lambda j.\neg\boldsymbol{B}(\boldsymbol{x},i,j))$

[2]In the second item from Definition 8.2.1.i, the 'updating' proposition is denoted by q (cf. the discussion of (4.2.20) in Ch. 4.2.4).

[3]Note the absence of a correlate (i.e. $q(j)$ for some intensional formula q) of the first conjunct from the definition of $q^\bullet$ in the definition of $x^\bullet$ in Definition 8.2.1.i.

$(\bigwedge \boldsymbol{x}.\boldsymbol{A})$	for	$(\lambda i\lambda j\forall \boldsymbol{x}.\boldsymbol{A}(i,j))$			
$(\bigvee \boldsymbol{x}.\boldsymbol{A})$	for	$(\lambda i\lambda j\exists \boldsymbol{x}.\boldsymbol{A}(i,j))$			
$\boldsymbol{B} \doteq \boldsymbol{C}$	for	$(\lambda \boldsymbol{x}\lambda i\lambda j.\boldsymbol{B}(\boldsymbol{x},i,j) = \boldsymbol{C}(\boldsymbol{x},i,j))$	$\top$	for	$(\lambda i\lambda j.\top)$
$(\boldsymbol{B} \wedge \boldsymbol{C})$	for	$(\lambda \boldsymbol{x}\lambda i\lambda j.\boldsymbol{B}(\boldsymbol{x},i,j) \wedge \boldsymbol{C}(\boldsymbol{x},i,j))$	$\boxdot \boldsymbol{A}$	for	$(\lambda i\lambda j.\Box \boldsymbol{A}(i))$
$(\boldsymbol{B} \vee \boldsymbol{C})$	for	$(\lambda \boldsymbol{x}\lambda i\lambda j.\boldsymbol{B}(\boldsymbol{x},i,j) \vee \boldsymbol{C}(\boldsymbol{x},i,j))$	$\diamond \boldsymbol{A}$	for	$(\lambda i\lambda j.\diamond \boldsymbol{A}(i))$

The derivations of the above definitions are included in Appendix C.3.

The constraint (C0) defines the STY_1^3 stand-in for the current situation, $\boldsymbol{w}$, via the strong single-type representation of the proposition denoted by $\lambda i\forall q.q(@) \twoheadrightarrow q(i)$. This representation is justified by the possibility of representing every situation via the set of all situations extending its propositional information (cf. the constraint (C0) from Ch. 7.2), by our 'strong' representation strategy for propositions (cf. Def. 8.2.1.i), and by the resulting equivalence of the defining TY_2^3 term from (C0) with the TY_2^3 term from (8.2.2) (below). The latter term designates a function from pairs of situations $\langle w_1, w\rangle$ to the truth-value of the proposition 'all propositions which are true at @ and w_1 and which carry information about one of the inhabitants of @ are also true at w'.

$$\lambda i\lambda j\forall p.\big((p(@) \wedge p(i)) \wedge (\exists x.E(x,@) \wedge abt(x,p))\big) \twoheadrightarrow p(j) \tag{8.2.2}$$

Like for the semantics of the logic WTY_1^3, the availability of a suitably typed representation of @ will allow a solution to Partee's temperature puzzle.

In correspondence with our representation of individuals in the type $(s\,s;t)$ (cf. Def. 8.2.1.i), the constraint (C3) defines the STY_1^3 constant ***bill*** as the designator of a function from pairs of situations $\langle w_1, w\rangle$ to the truth-value of the proposition 'all true propositions at w_1 which carry information about Bill are true at w' (i.e. as the designator of the characteristic function of the set of situation pairs, $\langle w_1, w\rangle$, with the above properties).

The remaining constraints enable the definition of the STY_1^3 translations of sentential PTQ forms as the designators of strong single-type representations of these forms' Montagovian (i.e. type-$(s;t)$) interpretations. Thus, the constraints (C4) to (C13) contribute to the STY_1^3 representation of propositions along the lines of (4.2.19). In particular, the constraint (C5) ensures the interpretation of the STY_1^3 constant ***walk*** as a function from propositional concepts $\boldsymbol{x}$ to the characteristic function of the set of pairs $\langle w_1, w\rangle$ at whose second element, w, the type-e correspondent, $\boldsymbol{x}^\circ$, of $\boldsymbol{x}$ walks and at which all $\boldsymbol{x}^\circ$-relevant true propositions whose designators are true at w_1 are true.

The semantic constraints on the interpretation of the $\mathcal{L}^{s1}$ constants ***bill*** and ***walk*** enable the definition of the STY_1^3 translation of the sentence Bill walks (in (8.2.3)). For better readability, we underline the definition of ***bill*** in step (3):

(8.2.3)

1. $[_{\text{NP}}\mathsf{Bill}] \rightsquigarrow \boldsymbol{bill} = \lambda i\lambda j\forall p.\big(p(i) \wedge abt(bill, p)\big) \twoheadrightarrow p(j)$
2. $[_{\text{VP}}[_{\text{IV}}\mathsf{walks}]] \rightsquigarrow \boldsymbol{walk} = \lambda \boldsymbol{x}\lambda i\lambda j.\, walk(\boldsymbol{x}^\circ, j) \wedge$
$\big(\forall p.(p(i) \wedge (\exists y.\, abt(y, [\lambda k.\, walk(\boldsymbol{x}^\circ, k)]) \wedge abt(y, p))) \twoheadrightarrow p(j)\big)$
3. $[_{\text{S}}[_{\text{NP}}\mathsf{Bill}][_{\text{VP}}[_{\text{IV}}\mathsf{walks}]]] \rightsquigarrow \boldsymbol{walk}(\boldsymbol{bill})$
$= \lambda \boldsymbol{x}\lambda i\lambda j.\, walk(\boldsymbol{x}^\circ, j) \wedge \big(\forall p.(p(i) \wedge (\exists y.\, abt(y, [\lambda k.\, walk(\boldsymbol{x}^\circ, k)]) \wedge$
$abt(y, p))) \twoheadrightarrow p(j)\big)\underline{\big[\lambda k_1\lambda k_2\forall q.\big(q(k_1) \wedge abt(bill, q)\big) \twoheadrightarrow q(k_2)\big]}$
$= \lambda i\lambda j.\, walk\big(\underline{\big[\lambda k_1\lambda k_2\forall q.\big(q(k_1) \wedge abt(bill, q)\big) \twoheadrightarrow q(k_2)\big]}^\circ, j\big) \wedge$
$\big(\forall p.\big(p(i) \wedge \big(\exists y.\, abt\big(y, \big[\lambda k.\, walk\big(\underline{\big[\lambda k_1\lambda k_2\forall q.\big(q(k_1) \wedge$
$\underline{abt(bill, q)\big) \twoheadrightarrow q(k_2)\big]}^\circ, k\big)\big]\big) \wedge abt(y, p)\big)\big) \twoheadrightarrow p(j)\big)$
$= \lambda i\lambda j.\, walk\big(bill, j\big) \wedge \big(\forall p.(p(i) \wedge (\exists y.\, abt(y, [\lambda k.\, walk(bill, k)]) \wedge abt(y, p))) \twoheadrightarrow p(j)\big)$
$= \lambda i\lambda j.\, walk\big(bill, j\big) \wedge \big(\forall p.(p(i) \wedge abt(bill, p)) \twoheadrightarrow p(j)\big)$

As expected, the term from the penultimate line of (8.2.3.3) results from applying the function $^\bullet$ to the intensional TY_2^3 formula $\lambda k.\, walk(bill, k)$. Since this formula only has a single type-e constituent, *bill* (s.t. $\forall y.\, abt(y, [\lambda k.\, walk(bill, k)]) \Rightarrow y = bill$, by **Ax14**–**Ax17**), the term from the penultimate line of (8.2.3.3) is equivalent to the simpler term from the last line of (8.2.3.3).

Admittedly, the definition of the STY_1^3 translation of the sentence Bill walks is much more complex than (a TY_2^3 equivalent of) the sentence's Montagovian translation (cf. (7.2.3.3)). However, since logical PTQ forms receive a translation into *simple* terms of the logic STY_1^3 (here, into the term $\boldsymbol{walk}(\boldsymbol{bill})$) – such that the terms' TY_2^3 definitions only describe the *underlying* semantic mechanisms –, the complexity of STY_1^3 definitions does not impact the adequacy or unificatory power of our STY_1^3-based single-type semantics. We will show in Section 8.4 that other attempts at providing a rich interpretation of names, cf. (**Landman, 1984**; **Scott, 1982**; **Merchant, 2008**), exhibit a similar representational complexity.

Like its counterpart in the logic WTY_1^3, Definition 8.2.2 enables us to define the STY_1^3 translations of all other lexical elements from Table 3.1. To show this, we obtain the definition of the STY_1^3 translation of the verb finds:

(8.2.4) $\lambda \boldsymbol{Q}\lambda \boldsymbol{x}.\boldsymbol{Q}(\lambda \boldsymbol{y}.\boldsymbol{find}(\boldsymbol{y}, \boldsymbol{x}))$
$= \lambda \boldsymbol{Q}\lambda \boldsymbol{x}.\boldsymbol{Q}\big(\lambda \boldsymbol{y}.\big[\lambda \boldsymbol{z}\lambda \boldsymbol{x}_1\lambda i\lambda j.\, find(\boldsymbol{z}^\circ, \boldsymbol{x}_1^\circ, j) \wedge \big(\forall p.(p(i) \wedge$
$(\exists z.\, abt(z, [\lambda k.\, find(\boldsymbol{z}^\circ, \boldsymbol{x}_1^\circ, k)]) \wedge abt(z, p))) \twoheadrightarrow p(j)\big)\big](\boldsymbol{y}, \boldsymbol{x})\big)$

$$= \lambda \boldsymbol{Q} \lambda \boldsymbol{x}. \boldsymbol{Q} \big(\lambda \boldsymbol{y} \lambda i \lambda j. \mathit{find}(\boldsymbol{y}^\circ, \boldsymbol{x}^\circ, j) \wedge \big(\forall p.(p(i) \wedge (\exists z. \mathit{abt}(z, [\lambda k. \mathit{find}(\boldsymbol{y}^\circ, \boldsymbol{x}^\circ, k)]) \wedge \mathit{abt}(z, p))) \twoheadrightarrow p(j)\big)\big)$$

The definitions of the STY_1^3 translation of the determiners a, every, and the are given in items (8.2.5) to (8.2.7). Like their counterparts in the logic WTY_1^3, these definitions involve the definitions of some of the designated single-type constants (here, the definitions of the STY_1^3 constants from Nota. 8.2.1):

(8.2.5) $$\begin{aligned} \mathsf{a} &\rightsquigarrow \lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \bigvee \boldsymbol{x}. \boldsymbol{P}_1(\boldsymbol{x}) \wedge \boldsymbol{P}(\boldsymbol{x}) \\ &= \lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \lambda i \lambda j \exists \boldsymbol{x}. \boldsymbol{P}_1(\boldsymbol{x}, i, j) \wedge \boldsymbol{P}(\boldsymbol{x}, i, j) \end{aligned}$$

(8.2.6) $$\begin{aligned} \mathsf{every} &\rightsquigarrow \lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \bigwedge \boldsymbol{x}. \boldsymbol{P}_1(\boldsymbol{x}) \dot{\rightarrow} \boldsymbol{P}(\boldsymbol{x}) \\ &= \lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \lambda i \lambda j \forall \boldsymbol{x}. \boldsymbol{P}_1(\boldsymbol{x}, i, j) \rightarrow \boldsymbol{P}(\boldsymbol{x}, i, j) \end{aligned}$$

(8.2.7) $$\begin{aligned} \mathsf{the} &\rightsquigarrow \lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}. (\boldsymbol{P}_1(\boldsymbol{y}) \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}) \wedge \boldsymbol{P}(\boldsymbol{x}) \\ &= \lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \lambda i \lambda j \exists \boldsymbol{x} \forall \boldsymbol{y}. (\boldsymbol{P}_1(\boldsymbol{y}, i, j) \leftrightarrow \boldsymbol{x} = \boldsymbol{y}) \wedge \boldsymbol{P}(\boldsymbol{x}, i, j) \end{aligned}$$

To show that our constraints on the STY_1^3 translations of lexical elements yield the 'right' definitions of the STY_1^3 translations of complex logical forms, we next define the STY_1^3 translations of the sentential forms from Part I, Chapter 3.2. We begin with the definition of the STY_1^3 translation of the sentence A man walks:

(8.2.8)

1. $[_{\mathrm{DET}}\mathsf{a}] \rightsquigarrow \lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \bigvee \boldsymbol{x}. \boldsymbol{P}_1(\boldsymbol{x}) \wedge \boldsymbol{P}(\boldsymbol{x})$
   $= \lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \lambda i \lambda j \exists \boldsymbol{x}. \boldsymbol{P}_1(\boldsymbol{x}, i, j) \wedge \boldsymbol{P}(\boldsymbol{x}, i, j)$
2. $[_{\mathrm{N}}\mathsf{man}] \rightsquigarrow \boldsymbol{man}$
   $= \lambda \boldsymbol{x} \lambda i \lambda j. \mathit{man}(\boldsymbol{x}^\circ, j) \wedge \big(\forall p.(p(i) \wedge (\exists y. \mathit{abt}(y, [\lambda k. \mathit{man}(\boldsymbol{x}^\circ, k)]) \wedge \mathit{abt}(y, p))) \twoheadrightarrow p(j)\big)$
3. $[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{man}]] \rightsquigarrow \lambda \boldsymbol{P} \bigvee \boldsymbol{x}. \boldsymbol{man}(\boldsymbol{x}) \wedge \boldsymbol{P}(\boldsymbol{x})$
   $= \lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \lambda i \lambda j \exists \boldsymbol{x}. \boldsymbol{P}_1(\boldsymbol{x}, i, j) \wedge \boldsymbol{P}(\boldsymbol{x}, i, j) \big[\lambda \boldsymbol{y} \lambda k \lambda k_1. \mathit{man}(\boldsymbol{y}^\circ, k_1) \wedge \big(\forall p.(p(k) \wedge (\exists y. \mathit{abt}(y, [\lambda k_2. \mathit{man}(\boldsymbol{x}^\circ, k_2)]) \wedge \mathit{abt}(y, p))) \twoheadrightarrow p(k_1)\big)\big]$
   $= \lambda \boldsymbol{P} \lambda i \lambda j \exists \boldsymbol{x}. \big(\mathit{man}(\boldsymbol{x}^\circ, j) \wedge \big(\forall p.(p(i) \wedge (\exists y. \mathit{abt}(y, p) \wedge \mathit{abt}(y, [\lambda k_2. \mathit{man}(\boldsymbol{x}^\circ, k_2)]))) \twoheadrightarrow p(j)\big)\big) \wedge \boldsymbol{P}(\boldsymbol{x}, i, j)$
4. $[_{\mathrm{VP}}[_{\mathrm{IV}}\mathsf{walks}]] \rightsquigarrow \boldsymbol{walks}$
   $= \lambda \boldsymbol{x} \lambda i \lambda j. \mathit{walk}(\boldsymbol{x}^\circ, j) \wedge \big(\forall p.(p(i) \wedge (\exists y. \mathit{abt}(y, [\lambda k. \mathit{walk}(\boldsymbol{x}^\circ, k)]) \wedge \mathit{abt}(y, p))) \twoheadrightarrow p(j)\big)$

5. $[_{\mathrm{S}}[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{man}]][_{\mathrm{VP}}[_{\mathrm{IV}}\mathsf{walks}]]] \rightsquigarrow \bigvee \boldsymbol{x}.\boldsymbol{man}(\boldsymbol{x}) \wedge \boldsymbol{walk}(\boldsymbol{x})$

$$
\begin{aligned}
&= \lambda \boldsymbol{P} \lambda i \lambda j \exists \boldsymbol{x}.\big(man(\boldsymbol{x}^{\circ}, j) \wedge \big(\forall p.(p(i) \wedge (\exists y.\, abt(y,p) \wedge abt(y, [\lambda k_2.\, man \\
&\qquad (\boldsymbol{x}^{\circ}, k_2)]))) \twoheadrightarrow p(j))\big) \wedge \boldsymbol{P}(\boldsymbol{x}, i, j)\big[\lambda \boldsymbol{y} \lambda k \lambda k_1.\, walk(\boldsymbol{y}^{\circ}, k_1) \wedge \\
&\qquad \big(\forall q.(q(k) \wedge (\exists z.\, abt(z,q) \wedge abt(z, [\lambda k_3.\, walk(\boldsymbol{y}^{\circ}, k_3)]))) \twoheadrightarrow q(k_1)\big)\big] \\
&= \lambda i \lambda j \exists \boldsymbol{x}.\big(man(\boldsymbol{x}^{\circ}, j) \wedge \big(\forall p.(p(i) \wedge (\exists y.\, abt(y,p) \wedge abt(y, \\
&\qquad [\lambda k_2.\, man(\boldsymbol{x}^{\circ}, k_2)]))) \twoheadrightarrow p(j))\big) \wedge \big(walk(\boldsymbol{x}^{\circ}, j) \wedge (\forall q.(q(i) \wedge \\
&\qquad (\exists z.\, abt(z,q) \wedge abt(z, [\lambda k_3.\, walk(\boldsymbol{x}^{\circ}, k_3)]))) \twoheadrightarrow q(j))\big) \\
&= \lambda i \lambda j \exists \boldsymbol{x}.\big(man(\boldsymbol{x}^{\circ}, j) \wedge walk(\boldsymbol{x}^{\circ}, j)\big) \wedge \big(\forall p.(p(i) \wedge (\exists y.\, abt(y,p) \wedge \\
&\qquad abt(y, [\lambda k_2.\, man(\boldsymbol{x}^{\circ}, k_2) \wedge walk(\boldsymbol{x}^{\circ}, k_2)])) \twoheadrightarrow p(j))\big) \\
&= \lambda i \lambda j \exists x.\big(man(x,j) \wedge walk(x,j)\big) \wedge \big(\forall p.\big(p(i) \wedge \big(\exists y.\, abt(y,p) \wedge \\
&\qquad abt\big(y, [\lambda k.\, man(x,k) \wedge walk(x,k)]\big)\big) \twoheadrightarrow p(j))\big)
\end{aligned}
$$

The definitions of the STY_1^3 translations of the forms from (3.2.3) to (3.2.27) are $^{\bullet}$-lifted variants of the TY_2^3 terms from (7.2.9) to (7.2.32). For illustration, some of these definitions are included below. Their derivations, and the definitions of the STY_1^3 translations of the remaining example sentences from (**Montague, 1973**), can be found in Appendix D.3.

(8.2.9) $[_{\mathrm{S}}[_{\mathrm{NP}}[\mathsf{the}][_{\mathrm{N}}\mathsf{man}]][_{\mathrm{VP}}[\mathsf{walks}]]] \rightsquigarrow \bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}.(\boldsymbol{man}(\boldsymbol{y}) \leftrightarrow \boldsymbol{x} \doteq \boldsymbol{y}) \wedge \boldsymbol{walk}(\boldsymbol{x})$

$$
\begin{aligned}
&= \lambda i \lambda j \exists x.\big(\forall y.(man(y,j) \leftrightarrow x = y) \wedge walk(x,j)\big) \wedge \big(\forall p.\big(p(i) \wedge \\
&\qquad \big(\exists z.\, abt(z,p) \wedge abt\big(z, \big[\lambda k.(\forall x_1.(man(x_1,k) \leftrightarrow x = x_1) \wedge \\
&\qquad\quad walk(x,k))\big]\big)\big) \twoheadrightarrow p(j)\big)\big)
\end{aligned}
$$

(8.2.10) $[_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{John}][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{finds}][_{\mathrm{NP}}\mathsf{Mary}]]] \rightsquigarrow \boldsymbol{find}(\boldsymbol{mary}, \boldsymbol{john})$

$$
\begin{aligned}
&= \lambda i \lambda j.\, find(mary, john, j) \wedge \\
&\qquad \big(\forall p.(p(i) \wedge (abt(mary,p) \vee abt(john,p))) \twoheadrightarrow p(j)\big)
\end{aligned}
$$

(8.2.11) $[_{\mathrm{S}}[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{unicorn}]]^0\, [_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{John}][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{seeks}]\ t_0]]]$

$$
\begin{aligned}
&\rightsquigarrow \bigvee \boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{seek}\big([\lambda \boldsymbol{P}.\boldsymbol{P}(\boldsymbol{x})], \boldsymbol{john}\big) \\
&= \lambda i \lambda j \exists x.\big(unicorn(x,j) \wedge seek([\lambda P \lambda k.\, P(x,k)], john, j)\big) \wedge \\
&\qquad \big(\forall p.(p(i) \wedge (abt(x,p) \vee abt(john,p))) \twoheadrightarrow p(j)\big)
\end{aligned}
$$

(8.2.12) $[_{S}[_{NP}\mathsf{John}][_{VP}[_{TV}\mathsf{seeks}][_{NP}[_{DET}\mathsf{a}][_{N}\mathsf{unicorn}]]]]$

$\rightsquigarrow\ \boldsymbol{seek}\big([\lambda\boldsymbol{P}\bigvee\boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x})\wedge\boldsymbol{P}(\boldsymbol{x})],\boldsymbol{john}\big)$

$=\ \lambda i\lambda j.seek\big([\lambda P\lambda k\exists x.unicorn(x,k)\wedge P(x,k)],john,j\big)\wedge$
$\big(\forall p.\big(p(i)\wedge\big(\exists y.abt(y,p)\wedge abt\big(y,\big[\lambda k_1.seek$
$\big([\lambda P\lambda k_2\exists x.unicorn(x,k_2)\wedge P(x,k_2)],john,k_1\big)\big]\big)\big)\twoheadrightarrow p(j)\big)$

(8.2.13) $[_{S}[_{NP}\mathsf{Bill}][_{VP}[_{TV}\mathsf{is}][_{NP}\mathsf{Mary}]]]\rightsquigarrow\boldsymbol{bill}\doteq\boldsymbol{mary}$
$=\lambda i\lambda j.bill=mary\wedge(\forall p.(p(i)\wedge(abt(bill,p)\vee abt(mary,p)))\twoheadrightarrow p(j))$

Our definition of the STY_1^3 translation of the sentence John does not find a unicorn (cf. (3.2.27)) justifies the replacement of the familiar constant for negation, $\neg$, by the constant for focused negation, $\sqsubset$, in the STY_1^3 translation of the adverb not:

(8.2.14) $[_{S}[_{NP}\mathsf{John}][_{VP}[\mathsf{does\ not}][_{VP}[_{TV}\mathsf{find}][_{NP}[_{DET}\mathsf{a}][_{N}\mathsf{unicorn}]]]]]$

$\rightsquigarrow\ \sqsubset\big(\bigvee\boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x})\wedge\boldsymbol{find}(\boldsymbol{x},\boldsymbol{john})\big)$

$=\ \lambda i\lambda j.\sqsubset\big([\exists x.unicorn(x,j)\wedge find(x,john,j)]\wedge$
$\big(\forall p.\big(p(i)\wedge\big(\exists y.abt\big(y,[\lambda k\exists x.unicorn(x,k)\wedge$
$find(x,john,k)]\big)\wedge abt\big(y,p\big)\big)\big)\big)\twoheadrightarrow p(j)\big)\big)$

$=\ \lambda i\lambda j.\big(\neg[\exists x.unicorn(x,j)\wedge find(x,john,j)]\wedge\big(\forall p.\big(p(i)\wedge\big(\exists y.abt\big(y,$
$\big([\lambda k.\neg(\exists x.unicorn(x,k)\wedge find(x,john,k))]\big)\wedge abt(y,p)\big)\big)\big)\twoheadrightarrow p(j)\big)\big)$

The STY_1^3 translation of not via the constant $\neg$ (cf. Def. 7.2.1) would assert the falsity of the designators of all i-true propositions which carry information about the aboutness subjects of the proposition denoted by $\lambda k\exists x.unicorn(x,k)\wedge find(x,john,k)$, rather than only the falsity of this proposition. But this contradicts our intuitions about the meaning of the sentence John does not find a unicorn.

Our semantic constraints on STY_1^3 terms give the expected definitions of the NP/CP-neutral verbs from (3.2.9) and (3.2.11) (in (8.2.15), resp. (8.2.20)):

(8.2.15) $[_{S}[_{NP}\mathsf{Pat}][_{VP}[_{TV}\mathsf{remembers}][_{NP}\mathsf{Bill}]]]\rightsquigarrow\boldsymbol{remember}(\boldsymbol{bill},\boldsymbol{pat})$
$=\ \lambda i\lambda j.remember(bill,pat,j)\wedge$
$\big(\forall p.(p(i)\wedge(abt(bill,p)\vee abt(pat,p)))\twoheadrightarrow p(j)\big)$

The definition of the 'strong' single-type translation of the logical form of (1b) (in (8.2.20), next but one page) illustrates the need for non-Montagovian type-e objects (and the resulting greater modeling power of single-type semantics w.r.t. traditional Montague semantics) even more forcefully than the form's 'weak' single-

type translation (cf. (7.2.16)). This is due to the fact that the first argument of the TY_2^3 term *remember* from (8.2.20) (in (8.2.16), below) has the form of a $^\bullet$-shifted version of an intensional formula (in (8.2.17)). But the identification of (8.2.17) with a type-$(s;t)$ term (in (8.2.18)) would violate the type requirement on the first argument of *remember*.

$$(8.2.16)\qquad \big[\lambda k_1\lambda k_2.\mathit{for}\big(\boldsymbol{x}_1^\circ, \mathit{wait}, \mathit{bill}, k_2\big) \wedge \big(\forall r.\big(p(k_1) \wedge \big(\exists x.\, \mathit{abt}\,\big(x, [\lambda k_3.\mathit{for}\,(\boldsymbol{x}_1^\circ, \mathit{wait}, \mathit{bill}, k_3)]\big) \wedge \mathit{abt}\,(x,r)\big)\big) \twoheadrightarrow r(k_2)\big)\big]^\circ$$

$$(8.2.17)\qquad = \big[\big(\lambda k_1.\mathit{for}\big(\boldsymbol{x}_1^\circ, \mathit{wait}, \mathit{bill}, k_1\big)\big)^\bullet\big]^\circ$$

$$(8.2.18)\qquad = \lambda k_1.\mathit{for}\big(\boldsymbol{x}_1^\circ, \mathit{wait}, \mathit{bill}, k_1\big)$$

To compensate for this problem, we 'flexibilize' the TY_2^3 definitions of basic-type STY_1^3 terms between type-$(s\,s;t)$ representations of individuals and propositions. Definition 8.2.3 demonstrates this possibility. In the definition, A is a TY_2^3 term of the type e:

DEFINITION 8.2.3. The operator *flex* (type $((s\,s;t)\,s\,s;t)$) renders the value of the function $^\circ$ flexible between type-e and -$(s;t)$ correlates of the TY_2^3 definitions of basic-type STY_1^3 terms. In particular, the operator *flex* behaves as follows:

$$\text{(a)}\quad \zeta\big(\mathit{flex}\big([A^\bullet]^\circ\big)\big) = \begin{cases} \zeta\big([A^\bullet]^\circ\big) & \text{if } \zeta \in T^2_{(e\,\alpha_1\ldots\alpha_n;t)} \\ \zeta\big(\big[\bigwedge_{w_1\in\mathcal{L}^2_s}[A^\bullet](w_1)\big]^\circ\big) & \text{if } \zeta \in T^2_{((s;t)\,\alpha_1\ldots\alpha_n;t)} \end{cases}$$

$$\text{(b)}\quad \zeta\big(\mathit{flex}\big([\varphi^\bullet]^\circ\big)\big) = \begin{cases} \zeta\big(\big[(\iota x.[\lambda k.E(x,k)] = \varphi)^\bullet\big]^\circ\big) & \text{if } \zeta \in T^2_{(e\,\alpha_1\ldots\alpha_n;t)} \\ \zeta\big([\varphi^\bullet]^\circ\big) & \text{if } \zeta \in T^2_{((s;t)\,\alpha_1\ldots\alpha_n;t)} \end{cases}$$

By the above, it holds that, if a TY_2^3 term of the form $[A^\bullet]^\circ$ serves as the argument of a type-$(\underline{(s;t)}\,\alpha_1\ldots\alpha_n;t)$ term, it is identified with the conjunction of the result of applying the TY_2^3 term $[A^\bullet]$ to each situation (cf. (a)). If a TY_2^3 term of the form $[\varphi^\bullet]^\circ$ serves as the argument of a type-$(\underline{e}\,\alpha_1\ldots\alpha_n;t)$ term, it is identified with the designator of the STY_1^3 representation of the type-e correspondent of the proposition φ (cf. (b)). Strategy (a) is motivated by the truth-evaluation of type-$(s\,s;t)$ terms from Section 8.3 (below). Strategy (b) is inspired by the strategy for the identification of individual proposition-correlates from Chapter 7.2.2 (cf. (7.2.16)).

In particular, the application of the *flex*-operator to the argument of the type-$(\underline{e}\,e\,s;t)$ constant *remember* (cf. (8.2.16)) yields the TY_2^3 term from the last line of (8.2.19):

$$(8.2.19)\quad \lambda x\lambda i.\mathit{remember}\big(\mathit{flex}\big(\big[\lambda k_1\lambda k_2.\mathit{for}\big(\boldsymbol{x}_1^\circ, \mathit{wait}, \mathit{bill}, k_2\big) \wedge \big(\forall r.\big(p(k_1) \wedge \big(\exists y.\, \mathit{abt}\,\big(y, [\lambda k.\mathit{for}\,(\boldsymbol{x}_1^\circ, \mathit{wait}, \mathit{bill}, k)]\big) \wedge \mathit{abt}\,(y,r)\big)\big) \twoheadrightarrow r(k_2)\big)\big]^\circ\big), x, i\big)$$

$$= \lambda x\lambda i.\mathit{remember}\big(\mathit{flex}\big(\big[(\lambda k.\mathit{for}\,(\boldsymbol{x}_1^\circ, \mathit{wait}, \mathit{bill}, k))^\bullet\big]^\circ\big), x, i\big)$$

(8.2.20)

1. $[_{\text{CP}}[_{\text{C}}\textsf{that}][_{\text{S}}[_{\text{NP}}\textsf{Bill}][_{\text{VP}}[_{\text{IV}}\textsf{waits}][_{\text{PP}}[_{\text{P}}\textsf{for}][_{\text{NP}}\textsf{she}_1]]]]] \rightsquigarrow \boldsymbol{for}(\boldsymbol{x}_1, \boldsymbol{wait}, \boldsymbol{bill})$
   $= \lambda i \lambda j.\mathit{for}\left(\boldsymbol{x}_1^{\circ}, \mathit{wait}, \mathit{bill}, j\right) \wedge \left(\forall p.\left(p(i) \wedge \left(\exists x.\,\mathit{abt}\left(x, [\lambda k.\mathit{for}\left(\boldsymbol{x}_1^{\circ}, \mathit{wait}, \mathit{bill}, k\right)]\right) \wedge \mathit{abt}\left(x, p\right)\right)\right) \twoheadrightarrow p(j)\right)$
2. $[_{\text{TV}}\textsf{remembers}] \rightsquigarrow \boldsymbol{remember}$
   $= \lambda \boldsymbol{y} \lambda \boldsymbol{x} \lambda i \lambda j.\mathit{remember}\left(\boldsymbol{y}^{\circ}, \boldsymbol{x}^{\circ}, j\right) \wedge \left(\forall p.\left(p(i) \wedge \left(\exists z.\,\mathit{abt}\left(z, p\right) \wedge \mathit{abt}\left(z, [\lambda k.\mathit{remember}\left(\boldsymbol{y}^{\circ}, \boldsymbol{x}^{\circ}, k\right)]\right)\right)\right) \twoheadrightarrow p(j)\right)$
3. $[_{\text{VP}}[_{\text{TV}}\textsf{remembers}][_{\text{CP}}[_{\text{C}}\textsf{that}][_{\text{S}}[_{\text{NP}}\textsf{Bill}][_{\text{VP}}[_{\text{IV}}\textsf{waits}][_{\text{PP}}[_{\text{P}}\textsf{for}][_{\text{NP}}\textsf{she}_1]]]]]]$
   $\rightsquigarrow \lambda \boldsymbol{x}.\,\boldsymbol{remember}\left(\boldsymbol{for}\left(\boldsymbol{x}_1, \boldsymbol{wait}, \boldsymbol{bill}\right), \boldsymbol{x}\right)$
   $= \lambda \boldsymbol{y} \lambda \boldsymbol{x} \lambda i \lambda j.\mathit{remember}\left(\boldsymbol{y}^{\circ}, \boldsymbol{x}^{\circ}, j\right) \wedge \left(\forall p.\left(p(i) \wedge \left(\exists z.\,\mathit{abt}\left(z, p\right) \wedge \mathit{abt}\left(z, [\lambda k.\mathit{remember}\left(\boldsymbol{y}^{\circ}, \boldsymbol{x}^{\circ}, k\right)]\right)\right)\right) \twoheadrightarrow p(j)\right)$
   $\underline{\left[\lambda k_1 \lambda k_2.\mathit{for}\left(\boldsymbol{x}_1^{\circ}, \mathit{wait}, \mathit{bill}, k_2\right) \wedge \left(\forall r.\left(p(k_1) \wedge \left(\exists x.\,\mathit{abt}\left(x, [\lambda k_3.\mathit{for}\left(\boldsymbol{x}_1^{\circ}, \mathit{wait}, \mathit{bill}, k_3\right)]\right) \wedge \mathit{abt}\left(x, r\right)\right)\right) \twoheadrightarrow r(k_2)\right)\right]}$
   $= \lambda \boldsymbol{x} \lambda i \lambda j.\mathit{remember}\left(\underline{\left[\lambda k_1 \lambda k_2.\mathit{for}\left(\boldsymbol{x}_1^{\circ}, \mathit{wait}, \mathit{bill}, k_2\right) \wedge \left(\forall r.\left(p(k_1) \wedge \left(\exists x.\,\mathit{abt}\left(x, [\lambda k_3.\mathit{for}\left(\boldsymbol{x}_1^{\circ}, \mathit{wait}, \mathit{bill}, k_3\right)]\right) \wedge \mathit{abt}\left(x, r\right)\right)\right) \twoheadrightarrow r(k_2)\right)\right]}^{\circ}, \boldsymbol{x}^{\circ}, j\right) \wedge \left(\forall p.\left(p(i) \wedge \left(\exists z.\,\mathit{abt}\left(z, p\right) \wedge \mathit{abt}\left(z, \left[\lambda k.\mathit{remember}\left(\underline{\left[\lambda k_1 \lambda k_2.\mathit{for}\left(\boldsymbol{x}_1^{\circ}, \mathit{wait}, \mathit{bill}, k_2\right) \wedge \left(\forall r.\left(p(k_1) \wedge \left(\exists x.\,\mathit{abt}\left(x, [\lambda k_3.\mathit{for}\left(\boldsymbol{x}_1^{\circ}, \mathit{wait}, \mathit{bill}, k_3\right)]\right) \wedge \mathit{abt}\left(x, r\right)\right)\right) \twoheadrightarrow r(k_2)\right)\right]}^{\circ}, \boldsymbol{x}^{\circ}, k\right)\right]\right)\right) \twoheadrightarrow p(j)\right)$
   $= \lambda \boldsymbol{x} \lambda i \lambda j.\mathit{remember}\left(\underline{\left[\lambda k_1.\mathit{for}\left(\boldsymbol{x}_1^{\circ}, \mathit{wait}, \mathit{bill}, k_1\right)\right)^{\bullet}\right]}^{\circ}, \boldsymbol{x}^{\circ}, j\right) \wedge \left(\forall p.\left(p(i) \wedge \left(\exists z.\,\mathit{abt}\left(z, p\right) \wedge \mathit{abt}\left(z, \left[\lambda k.\mathit{remember}\left(\left(\underline{\left[\lambda k_1.\mathit{for}\left(\boldsymbol{x}_1^{\circ}, \mathit{wait}, \mathit{bill}, k_1\right)\right)^{\bullet}\right]}^{\circ}, \boldsymbol{x}^{\circ}, k\right)\right]\right)\right)\right) \twoheadrightarrow p(j)\right)$
4. $[_{\text{S}}\ t_1\ [_{\text{VP}}[_{\text{TV}}\textsf{remembers}][_{\text{CP}}[_{\text{C}}\textsf{that}][_{\text{S}}[_{\text{NP}}\textsf{Bill}][_{\text{VP}}[_{\text{IV}}\textsf{waits}][_{\text{PP}}[_{\text{P}}\textsf{for}][_{\text{NP}}\textsf{she}_1]]]]]]]$
   $\rightsquigarrow \boldsymbol{remember}\left(\boldsymbol{for}\left(\boldsymbol{x}_1, \boldsymbol{wait}, \boldsymbol{bill}\right), \boldsymbol{x}_1\right)$
   $= \lambda \boldsymbol{x} \lambda i \lambda j.\mathit{remember}\left(\underline{\left[\lambda k_1.\mathit{for}\left(\boldsymbol{x}_1^{\circ}, \mathit{wait}, \mathit{bill}, k_1\right)\right)^{\bullet}\right]}^{\circ}, \boldsymbol{x}^{\circ}, j\right) \wedge \left(\forall p.\left(p(i) \wedge \left(\exists z.\,\mathit{abt}\left(z, p\right) \wedge \mathit{abt}\left(z, \left[\lambda k.\mathit{remember}\left(\left(\underline{\left[\lambda k_1.\mathit{for}\left(\boldsymbol{x}_1^{\circ}, \mathit{wait}, \mathit{bill}, k_1\right)\right)^{\bullet}\right]}^{\circ}, \boldsymbol{x}^{\circ}, k\right)\right]\right)\right)\right) \twoheadrightarrow p(j)\right)[\boldsymbol{x}_1]$

$= \lambda i \lambda j.remember\big(\underline{\big[\lambda k_1.for\big(\boldsymbol{x}_1^\circ, wait, bill, k_1\big)\big)^\bullet\big]^\circ}, \boldsymbol{x}_1^\circ, j\big) \wedge \big(\forall p.\big(p(i) \wedge \big(\exists z.\, abt\,(z,p) \wedge abt\,\big(z, \big[\lambda k.\, remember$
$\big(\big(\underline{\big[\lambda k_1.for\big(\boldsymbol{x}_1^\circ, wait, bill, k_1\big)\big)^\bullet\big]^\circ}, \boldsymbol{x}_1^\circ, k\big)\big]\big)\big)\big) \twoheadrightarrow p(j)\big)$

5. $[_S[_{NP}\mathsf{Pat}]^1\,[_S\ t_1\ [_{VP}[_{TV}\mathsf{remembers}][_{CP}[_C\mathsf{that}][_S[_{NP}\mathsf{Bill}][_{VP}[_{IV}\mathsf{waits}][_{PP}[_P\mathsf{for}][_{NP}\mathsf{she}_1]]]]]]]]$

$\rightsquigarrow \boldsymbol{remember}\,(\boldsymbol{for}\,(\boldsymbol{pat}, \boldsymbol{wait}, \boldsymbol{bill}), \boldsymbol{pat})$

$= \lambda \boldsymbol{x}_1 \lambda i \lambda j.remember\big(\underline{\big[\lambda k_1.for\big(\boldsymbol{x}_1^\circ, wait, bill, k_1\big)\big)^\bullet\big]^\circ}, \boldsymbol{x}_1^\circ, j\big) \wedge \big(\forall p.\big(p(i) \wedge \big(\exists z.\, abt\,(z,p) \wedge abt\,\big(z, \big[\lambda k.\, remember$
$\big(\big(\underline{\big[\lambda k_1.for\,\big(\boldsymbol{x}_1^\circ, wait, bill, k_1\big)\big)^\bullet\big]^\circ}, \boldsymbol{x}_1^\circ, k\big)\big]\big)\big)\big) \twoheadrightarrow p(j)\big)\ \big[\lambda k_2 \lambda k_3\,\big(\forall q.p(k_2) \wedge abt\,(q, pat) \twoheadrightarrow q(k_3)\big)\big]$

$= \lambda i \lambda j.remember\big(\underline{\big[\lambda k_1.for\,\big(\big[\lambda k_2 \lambda k_3\,\big(\forall q.p(k_2) \wedge abt\,(q, bill) \twoheadrightarrow q(k_3)\big)\big]^\circ, wait, bill, k_1\big)\big)^\bullet\big]^\circ}, \big[\lambda k_2 \lambda k_3\,\big(\forall q.p(k_2) \wedge$
$abt\,(q, pat) \twoheadrightarrow q(k_3)\big)\big]^\circ, j\big) \wedge \big(\forall p.\big(p(i) \wedge \big(\exists z.\, abt\,(z,p) \wedge abt\,\big(z, \big[\lambda k.\, remember\,\big(\big(\underline{\big[\lambda k_1.for\,\big(\big[\lambda k_2 \lambda k_3\,\big(\forall q.p(k_2) \wedge$
$abt\,(q, bill) \twoheadrightarrow q(k_3)\big)\big]^\circ, wait, bill, k_1\big)\big)^\bullet\big]^\circ}, \big[\lambda k_2 \lambda k_3\,\big(\forall q.p(k_2) \wedge abt\,(q, pat) \twoheadrightarrow q(k_3)\big)\big]^\circ, k\big)\big]\big)\big)\big) \twoheadrightarrow p(j)\big)$

$= \lambda i \lambda j.remember\big(\underline{\big[\lambda k_1.for\,\big(\big[\lambda k_2 \lambda k_3\,\big(\forall q.p(k_2) \wedge abt\,(q, bill) \twoheadrightarrow q(k_3)\big)\big]^\circ, wait, bill, k_1\big)\big)^\bullet\big]^\circ}, pat, j\big) \wedge \big(\forall p.\big(p(i) \wedge$
$\big(\exists z.\, abt\,(z,p) \wedge abt\,\big(z, \big[\lambda k.\, remember\,\big(\big(\underline{\big[\lambda k_1.for\,\big(\big[\lambda k_2 \lambda k_3\,\big(\forall q.p(k_2) \wedge abt\,(q, bill) \twoheadrightarrow q(k_3)\big)\big]^\circ, wait, bill, k_1\big)\big)^\bullet\big]^\circ},$
$pat, k\big)\big]\big)\big)\big) \twoheadrightarrow p(j)\big)$

$$= \lambda x \lambda i.\, remember\big(\big[\big(\iota y.(\lambda k.E(y,k)) = [\lambda k_1.for(\boldsymbol{x}_1^{\circ}, wait, bill, k_1)]\big)^{\bullet}\big]^{\circ}, x, i\big)$$
$$= \lambda x \lambda i.\, remember\big(\big[\iota y.(\lambda k.E(y,k)) = [\lambda k_1.for(\boldsymbol{x}_1^{\circ}, wait, bill, k_1)]\big], x, i\big)$$

The STY_1^3 translation of the LF from (3.2.11) can then be defined as follows:

(8.2.21)

$[_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{Pat}]^1\ [_{\mathrm{S}}\ t_1\ [_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{remembers}][_{\mathrm{CP}}[_{\mathrm{C}}\mathsf{that}][_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{Bill}][_{\mathrm{VP}}[_{\mathrm{IV}}\mathsf{waits}][_{\mathrm{PP}}[_{\mathrm{P}}\mathsf{for}][_{\mathrm{NP}}\mathsf{she}_1]]]]]]]]$

$$\rightsquigarrow \boldsymbol{remember}\big(\boldsymbol{for}(\boldsymbol{pat}, \boldsymbol{wait}, \boldsymbol{bill}), \boldsymbol{pat}\big)$$
$$= \lambda i \lambda j.\, remember\big([\iota y.[\lambda k.for(pat, wait, bill, k)] = (\lambda k_1.E(y,k_1))], pat, j\big) \wedge$$
$$\big(\forall p.\big(p(i) \wedge \big(\exists x.\, abt\big(x, \big[\lambda k_2.\, remember\big([\iota y.[\lambda k.for(pat, wait, bill, k)] =$$
$$(\lambda k_1.E(y,k_1))], pat, k_2\big)\big]\big) \wedge abt(x,p)\big)\big) \twoheadrightarrow p(j)\big)$$
$$= \lambda i \lambda j.\, remember\big([\iota y.[\lambda k.for(pat, wait, bill, k)] = (\lambda k_1.E(y,k_1))], pat, j\big) \wedge$$
$$\big(\forall p.\big(p(i) \wedge (abt(pat,p) \vee abt(bill,p))\big) \twoheadrightarrow p(j)\big)$$

Expectedly, the *flex*-operator is also instrumental in our definition of the STY_1^3 translation of CP equatives. The definition of the STY_1^3 translation of (3.2.22.5) is given below. There, X abbreviates $x = [\iota z.(\lambda k_1.hate(bill, mary, k_1)) = (\lambda k_2.E(z,k_2))]$:

(8.2.22)

$[_{\mathrm{S}}[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{the}][_{\mathrm{N}}\mathsf{problem}]][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{is}][_{\mathrm{CP}}[_{\mathrm{C}}\mathsf{that}][_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{Mary}][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{hates}][_{\mathrm{NP}}\mathsf{Bill}]]]]]]$

$$\rightsquigarrow \bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}.(\boldsymbol{problem}(\boldsymbol{y}) \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}) \wedge \boldsymbol{x} \doteq \boldsymbol{hate}(\boldsymbol{bill}, \boldsymbol{mary})$$
$$= \lambda i \lambda j \exists x \forall y.((problem(y,j) \leftrightarrow x = y) \wedge X) \wedge \big(\forall p.\big(p(i) \wedge$$
$$\big(\exists z.\, abt\big(z, [\lambda k.(problem(y,k) \leftrightarrow x = y) \wedge X]\big) \wedge abt(z,p)\big)\big) \twoheadrightarrow p(j)\big)$$

Since the TY_2^3 type for individuals is not conjoinable, we are unable to define the STY_1^3 translations of coordinations with a proper name- and a CP conjunct (cf. Ch. 7.2.2). To circumvent this problem, we translate the form from (7.2.34) (or (7.2.35)) instead of the form from (3.2.25). The STY_1^3 translation of (7.2.34) is defined as follows:

(8.2.23)

$[_{\mathrm{S}}[\mathsf{Pat}]^1\ [_{\mathrm{S}}\ t_1\ [_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{remembers}][_{\mathrm{NP}}\mathsf{Bill}]]$
$[[_{\mathrm{CONJ}}\mathsf{and}][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{remembers}][_{\mathrm{CP}}[_{\mathrm{C}}\mathsf{that}][_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{Bill}][_{\mathrm{VP}}[_{\mathrm{IV}}\mathsf{waits}][_{\mathrm{PP}}[_{\mathrm{P}}\mathsf{for}][_{\mathrm{NP}}\mathsf{she}_1]]]]]]]]]$

$$\rightsquigarrow \big(\lambda \boldsymbol{x}.\, \boldsymbol{remember}(\boldsymbol{bill}, \boldsymbol{x}) \wedge \boldsymbol{remember}\big(\boldsymbol{for}(\boldsymbol{x}, \boldsymbol{wait}, \boldsymbol{bill}), \boldsymbol{x}\big)\big)(\boldsymbol{pat})$$
$$= \big(\lambda x \lambda i \lambda j.\, remember\big([\iota y.[\lambda k.for(x, wait, bill, k)] = (\lambda k_1.E(y,k_1))], x, j\big) \wedge$$
$$\big(\forall p.\big(p(i) \wedge (abt(x,p) \vee abt(bill,p))\big) \twoheadrightarrow p(j)\big)\big)(pat)$$

$$= \lambda i\lambda j.\mathit{remember}\big(\big[\iota y.[\lambda k.\mathit{for}(\mathit{pat},\mathit{wait},\mathit{bill},k)] = (\lambda k_1.E(y,k_1))\big],\mathit{pat},j\big) \wedge \big(\forall p.\big(p(i) \wedge (\mathit{abt}(\mathit{pat},p) \vee \mathit{abt}(\mathit{bill},p))\big) \twoheadrightarrow p(j)\big)$$

The possibility of defining the STY_1^3 translation of (an intuitive equivalent of) sentence (4) is in agreement with a strong single-type version of Proposition 3.2:

PROPOSITION 8.1 (NP/CP coordinability). *A strong single-type semantics can model logical forms which contain coordinations with a proper name- and a* CP*-conjunct.*

This completes our definition of the STY_1^3 translations of logical PTQ forms. To assess our semantics' ability to accommodate the phenomena from Proposition 1.3, we next specify the truth-conditions for logical forms of the basic STY_1^3 type, and identify their sentential equivalents.

8.3. PTQ Truth and Equivalence

Like in the semantics of the logic WTY_1^3, the notions of STY_1^3-based truth and falsity extend from the logical forms of sentences to proper names. However, since the logic STY_1^3 does not command the propositional type $(s;t)$, the truth (or falsity) of names and sentences cannot be obtained via the truth (resp. falsity) of the definitions of these forms' STY_1^3 translations (w.r.t. the designated model of the logic TY_2^3).

To compensate for this shortcoming, we determine the truth of sentences via the truth of the *conjunction* of the results of applying the definition of the sentences' STY_1^3 translation to the designator of each situation in the set W. Since we know (by a strong[4] reading of **Ax12**) that there is, for each individual, at least one situation in which this individual does not exist, we can identify this conjunction with the updating intensional TY_2^3 formula in the definition of the sentences' STY_1^3 translation. For the sentence Barbara Partee arrives, this conjunction is given in (8.3.1):

(8.3.1)
$$\bigwedge_{w_1 \in \mathcal{L}_s^2} \big(\lambda j.\mathit{arrive}(\mathit{partee},j) \wedge (\forall p.(p(w_1) \wedge \mathit{abt}(\mathit{partee},p)) \twoheadrightarrow p(j))\big)$$
$$= \bigwedge_{w_1 \in \mathcal{L}_s^2} \big(\lambda j.[\lambda i.\mathit{arrive}(\mathit{partee})]^{\bullet}(w_1,j)\big)$$
$$= \lambda i.\mathit{arrive}(\mathit{partee})$$

[4]This reading exploits the discernibility of non-identical individuals (cf. (6.1.1)) and the possibility of forming (the individual domain of) 'new' situations by identifying the individuals which witness a given property at another situation (cf. Ch. 6.1). It is further supported by Kratzer's (**2011**) assumption of minimal situations at which only a single atomic sentence is true.

Since the type-$(s\,s;t)$ representation of individuals does not express an update on the available information at the relevant situation, the evaluation strategy from (8.3.1) is not available for *proper names*. To compensate for this shortcoming, we determine the truth of names via the truth of the result of applying the definition of the name's STY_1^3 translation to the designator of some *representing* (or *local*) situation (e.g. the information state $@'$ at which some cognitive agent obtains the rich propositional representation of the name's type-e referent). For the name Barbara Partee and the representing situation $@'$, the resulting term is given in (8.3.2):

$$\lambda j \forall p.(p(@') \wedge abt(partee, p)) \twoheadrightarrow p(j) \tag{8.3.2}$$

Thus, to evaluate the truth (or falsity) of the name Barbara Partee at the current situation $@$, we check whether the conjunction of the designators of all true propositions at the state $@'$ which carry information about Barbara Partee is true (resp. false) at $@$. As a result of this strategy, names in STY_1^3 semantics no longer share the truth-conditions of their containing simple existential sentences.

The notion of STY_1^3-based linguistic truth is defined below. In the definition, we let $\boldsymbol{A}_{(s\,s;t)}$ be the STY_1^3 translation of some logical sentence form X, and let $\boldsymbol{B}_{(s\,s;t)}$ be the STY_1^3 translation of some proper name Y, s.t. $X \rightsquigarrow \boldsymbol{A}$ and $Y \rightsquigarrow \boldsymbol{B}$. We let M^2 and $M_{\mathcal{F}^{s1}}$ be the designated models of the logics TY_2^3, resp. STY_1^3, and let g^2 and $g^{s1} = g^{2\restriction \mathsf{s1Type}}$ be their assignments. We assume that w and $@'$ are the evaluating and the representing situation, respectively:

Definition 8.3.1 (STY_1^3-based linguistic truth). A logical sentence form X is *true* (or *false*) at w in M^2 under g^2, i.e. $\textsc{true}_{M^{s1},w}(X)$ (resp. $\textsc{false}_{M^{s1},w}(X)$), if $w \models_{M^2} \bigwedge_{w_1 \in \mathcal{L}_s^2}(\lambda j.\boldsymbol{A}(w_1, j))$ (resp. $w \nvDash_{M^2} \bigwedge_{w_1 \in \mathcal{L}_s^2}(\lambda j.\boldsymbol{A}(w_1, j))$).

The $@'$-specific representation of the proper name Y is *true* (or *false*) at w in M^2 under g^2, i.e. $\textsc{true}_{M^{s1},w}(Y)$ (resp. $\textsc{false}_{M^{s1},w}(Y)$), if $w \models_{M^2} (\lambda j.\boldsymbol{B}(@', j))$ (resp. $w \nvDash_{M^2} (\lambda j.\boldsymbol{B}(@', j))$).

As anticipated, Definition 8.3.1 yields the familiar truth-conditions for the logical forms of sentences. The truth- and falsity-conditions of the sentence Barbara Partee arrives are given below:

$$\begin{aligned}
&\textsc{true}_{M^{s1},@}\big([_{\mathrm{S}}[_{\mathrm{NP}}\textsf{Barbara Partee}][_{\mathrm{VP}}[_{\mathrm{IV}}\textsf{arrives}]]]\big) \qquad (8.3.3)\\
&\text{iff } @ \models_{M^2} \bigwedge_{w_1 \in \mathcal{L}_s^2} \big(\lambda j.\boldsymbol{arrive}(\boldsymbol{partee})(w_1, j)\big)\\
&\text{iff } @ \models_{M^2} \bigwedge_{w_1 \in \mathcal{L}_s^2} \big(\lambda j.arrive(partee, j) \wedge \\
&\qquad\qquad (\forall p.(p(w_1) \wedge abt(partee, p)) \twoheadrightarrow p(j))\big)\\
&\text{iff } @ \models_{M^2} \lambda i.arrive(partee, i)
\end{aligned}$$

(8.3.4)
$$\begin{aligned}
&\textsc{false}_{M^{s1},@}\big([_{S}[_{NP}\textsf{Barbara Partee}][_{VP}[_{IV}\textsf{arrives}]]]\big)\\
\text{iff}\ \ &@ \dashv\!\models_{M^2} \bigwedge_{w_1\in\mathcal{L}^2_s} \big(\lambda j.\,\boldsymbol{arrive}(\boldsymbol{partee})(w_1,j)\big)\\
\text{iff}\ \ &@ \models_{M^2} \bigwedge_{w_1\in\mathcal{L}^2_s} \big(\lambda j.\,\neg\boldsymbol{arrive}(\boldsymbol{partee})(w_1,j)\big)\\
\text{iff}\ \ &@ \models_{M^2} \bigwedge_{w_1\in\mathcal{L}^2_s} \big(\lambda j.\,\neg arrive(partee,j)\wedge\\
&\qquad\qquad (\forall p.(p(w_1)\wedge abt(partee,p))\twoheadrightarrow p(j))\big)\\
\text{iff}\ \ &@ \models_{M^2} \lambda i.\,\neg arrive(partee,i)
\end{aligned}$$

Definition 8.3.1 also yields the intuitive truth- and falsity-conditions for proper names. For the isolated utterance of the name Barbara Partee in the context of the representing situation $@'$, these conditions are given below:

(8.3.5)
$$\begin{aligned}
&\textsc{true}_{M^{s1},@}\big([_{NP}\textsf{Barbara Partee}]\big)\\
\text{iff}\ \ &@ \models_{M^2} \lambda j.\,\boldsymbol{partee}(@',j)\\
\text{iff}\ \ &@ \models_{M^2} \lambda j\forall p.\big(p(@')\wedge abt(partee,p)\big)\twoheadrightarrow p(j)
\end{aligned}$$

(8.3.6)
$$\begin{aligned}
&\textsc{false}_{M^{s1},@}\big([_{NP}\textsf{Barbara Partee}]\big)\\
\text{iff}\ \ &@ \dashv\!\models_{M^2} \lambda j.\,\boldsymbol{partee}(@',j)\\
\text{iff}\ \ &@ \dashv\!\models_{M^2} \lambda j\forall p.\big(p(@')\wedge abt(partee,p)\big)\twoheadrightarrow p(j)
\end{aligned}$$

The STY^3_1 truth- and falsity-conditions from (8.3.5) and (8.3.6) formalize the intuitive truth- and falsity-conditions for proper names from Chapter 1.2.1. In particular, if, in an agent's representing situation $@'$, Barbara Partee has the property of arriving, then, since it holds for the situation @ from (9) that $@ \models_{M^2} (\lambda i.\ arrive(partee,i))$, the $@'$-specific representation of the name Barbara Partee is evaluated 'true' at @. If, in an agent's representing situation $@'$, Barbara Partee has the property of *not* arriving, then the $@'$-specific representation of the name Barbara Partee is evaluated 'false' at @.

Remarkably, since we have identified evaluating indices with *partial situations*, it may happen that the conjunction of the designators of all true propositions at the state $@'$ which carry information about Barbara Partee is undefined at @. This observation is captured below:

Proposition 8.2 (STY^3_1 truth-aptness of names). *In a strong single-type semantics, proper names receive a partial truth-value at some situations.*

Our interpretation of proper names in strong single-type semantics also enables the non-trivial equivalence of the interpretation of logical forms at a situa-

tion. The resulting relation of semantic equivalence is defined in reference to the *local equivalence* relation between TY_2^3 terms (cf. Def. 6.4.4).

In the definition of linguistic equivalence, we let $\boldsymbol{A}_{(s\,s;t)}$ and $\boldsymbol{B}_{(s\,s;t)}$ be the STY_1^3 translations of the logical forms X, respectively Y, such that $X \rightsquigarrow \boldsymbol{A}$ and $Y \rightsquigarrow \boldsymbol{B}$. We assume that $@'$ is the representing situation.

Definition 8.3.2 (Local STY_1^3-based linguistic equivalence). A logical form X is *locally equivalent* to a form Y at $@'$ (or is $@'$-*equivalent* to Y) in M^{s1} under g^{s1}, i.e. $@'$-$\mathrm{MEANS}_{M^{s1}}(Y, X)$, if $\models_g (\lambda i.\, \boldsymbol{A}(@', i)) = (\lambda i.\, \boldsymbol{B}(@', i))$ in M^2 under g^2.

The notion of local STY_1^3-based equivalence supports the situation-specific equivalence of proper names and *contextually salient* sentences. Thus, at every situation w at which the intensional formula $\lambda i.\, \mathit{arrive}\,(\mathit{partee}, i)$ is true, the name Barbara Partee will be locally equivalent to the sentence Barbara Partee arrives:

$$
\begin{aligned}
(8.3.7)\qquad & w\text{-}\mathrm{MEANS}_{M^{s1}}\big([_{\mathrm{NP}}\text{Barbara Partee}], [_{\mathrm{S}}[_{\mathrm{NP}}\text{Barbara Partee}][_{\mathrm{VP}}[_{\mathrm{IV}}\text{arrives}]]]\big)\\
\text{iff}\quad & \models_{M^2} (\lambda i.\, \boldsymbol{partee}\,(w, i)) = (\lambda i.\, \boldsymbol{arrive}\,(\boldsymbol{partee})(w, i))\\
\text{iff}\quad & \models_{M^2} \big(\lambda i \forall p.\big(p(w) \wedge \mathit{abt}\,(\mathit{partee}, p)\big) \twoheadrightarrow p(i)\big)\\
=\quad & \big(\lambda i.\, \mathit{arrive}\,(\mathit{partee}, i) \wedge \big(\forall p.\big(p(w) \wedge \mathit{abt}\,(\mathit{partee}, p)\big) \twoheadrightarrow p(i)\big)\big)
\end{aligned}
$$

Our definition of local STY_1^3-based equivalence models the attested equivalence relations between proper names and sentences from Chapter 1.2.1.

The above motivates our description of STY_1^3-based semantics as a *strong* single-type semantics for the PTQ fragment, which accommodates Proposition 1.3.ii. For the designated model of the logic STY_1^3, this proposition is restated below:

Proposition 8.3 (Contextually salient STY_1^3 name equivalents). *In a strong single-type semantics, proper names can stand in equivalence relations to contextually salient non-existential sentences. Instances of this relation are identifiable via the local equivalence of the* TY_2^3 *definitions of the forms'* STY_1^3 *translations.*

To obtain situation-*general* equivalence and entailment relations between logical forms in our strong single-type semantics, we use the notions of *global* TY_2^3 equivalence and entailment. As a result, general equivalence and entailment in our strong single-type semantics are defined in analogy to the corresponding relations in the semantics of the logic WTY_1^3 (cf. Def. 7.3.2, 7.3.3).

Notably, in contrast to the relation of global equivalence on logical forms in the semantics of the logic WTY_1^3, the global equivalence relation in our strong STY_1^3-based semantics does not obtain between proper names and sentences. This is due to the fact that the definitions of the STY_1^3 translations of proper names do not express an update on the available information at the relevant situation, and to our assumption that each individual fails to exist in *some* situation (cf. the strong reading of **Ax12**). The resulting observation is captured below:

PROPOSITION 8.4 (Absence of global STY_1^3 name equivalents). *In a strong single-type semantics, proper names do not have global sentential equivalents.*

Proposition 8.4 reflects the intuitive absence of a name's sentential equivalent(s) independent of a specific context. Moreover, it avoids the negative consequences of a single-type semantics for the semantic motivation of the syntactic category system (cf. Ch. 3.4). In particular, since the definitions of the STY_1^3 translations of simple existential sentences do *not* satisfy the predicate for the description of the STY_1^3 sort for individuals (in (8.3.8)), our STY_1^3-based semantics will *not* classify the STY_1^3 translation of the sentence Bill exists as an expression of the sort $(s\,s;t)_\imath$ (where $\imath$ abbreviates the predicate from (8.3.8)). As a result, this semantics *will* be able to explain the well-formedness of the logical form from (23), and the ill-formedness of the expressions from (24a) and (24b) (cf. Ch. 7.4).

(8.3.8) $\lambda\pi.\left[\exists x.\, mont(x) \wedge \pi = (\lambda i \lambda j \forall p.(p(i) \wedge abt(x,p)) \twoheadrightarrow p(j))\right]$

(8.3.9) $\lambda\pi.\left[\exists q.\, \pi = \big(\lambda i \lambda j.q(j) \wedge (\forall p.(p(i) \wedge (\exists y.\, abt(y,q) \wedge abt(y,p))) \twoheadrightarrow p(j))\big)\right]$

The close correspondence between syntactic categories and STY_1^3 subtypes is captured in Figure 8.1. In the figure, the sort label ρ abbreviates the predicate from (8.3.9). In (8.3.8) and (8.3.9), the variable π ranges over type-$(s\,s;t)$ objects.

(GB) Syntax	S	NP	N	C	SAV	⋯
STY_1^3 Semantics (Object theory)	$(s\,s;t)_\rho$	$(s\,s;t)_\imath$	⋮	$((s\,s;t)_\rho\, s\, s;t_\rho)$		⋯

FIGURE 8.1. Syntactic categories and STY_1^3 subtypes.

This completes our discussion of the linguistic truth- and equivalence-relations in the semantics of the logic STY_1^3. We close the chapter with a defense of the semantic complexity of our strong single-type objects.

8.4. A Note on Semantic Complexity

In Section 8.2.2, we have observed that our semantic constraints on the STY_1^3 translations of lexical PTQ elements display a high syntactic complexity. Thus, the definition of the STY_1^3 translation of the name Bill (in (8.4.1), next page) consists of a significantly 'longer' lambda term than the definition of the name's translation into a term of the logic WTY_1^3 (in (8.4.2)) or of the logic TY_2^3 (in (8.4.3)):

$$\lambda i \lambda j \forall p.(p(i) \wedge abt(bill, p)) \twoheadrightarrow p(j) \tag{8.4.1}$$

$$\lambda i.E(bill, i) \tag{8.4.2}$$

$$bill \tag{8.4.3}$$

Since the definitions of all other STY_1^3 translations are similarly complex, the derivation of the definitions of STY_1^3 translations of non-lexical logical forms (e.g. the derivation of the definition of the STY_1^3 term $\boldsymbol{walk}(\boldsymbol{bill})$ from (8.2.3)) require many more lambda conversions than their counterparts in the logics WTY_1^3 (cf. (7.2.3)) or TY_2^3. From a practitioner's point of view, this makes our strong single-type semantics much less appealing than its weaker counterpart, and than the formal semantics from (**Montague, 1973**).

The present section defends the high semantic complexity of our strong single-type objects. In particular, it argues that this complexity is demanded by the possibility of representing worldly objects in terms of their available *information* (cf. Ch. 4.2.4), and of accommodating the assertoric interpretation of names from Proposition 1.3.ii (cf. Ch. 1.2.1). To support this claim, we cite a number of semantic theories which also postulate complex values for names. It is demonstrated that the formalization of these theories and their generalization across situations yields at least equally complex objects as our semantics. In fact, our strong single-type semantics even proves to be the *simplest* theory which models these phenomena.

Below, we first show the high complexity of the semantic values of names in Landman's version of Data Semantics (**Landman, 1984**), cf. (**Scott, 1982**). We will then do the same for Merchant's semantic account of ellipsis (**Merchant, 2008**). We close the section with a comparison between our single-type semantics and propositional logics with only the Sheffer stroke.

8.4.1. Interpretations of Names in Data Semantics. In (**Landman, 1984**), cf. (**Landman, 1986**), Landman provides a semantic theory of partial information which captures the incomplete knowledge of linguistic agents about worldly objects. The theory is shown to solve a number of problems regarding the failure of substitutivity of logically equivalent expressions in epistemic contexts.

To solve these problems, Landman replaces possible worlds by partial information states. Such states are proper filters in the set of propositions, which contain the propositions that are implicitly assumed in conversation. At each state, proper names are interpreted as subsets of the state whose members carry true information about the name's referent at that state, cf. (**Scott, 1982**, p. 579–584). Below, we let 0 be the constant for the absurd proposition, and let $\circ$ denote the operation of information combination. Landman's description of the referent of Hesperus at the state σ from (**Landman, 1984**) can then be formalized as follows,

where *hesp* is the constant for Hesperus, and where p, q, and r are variables over primitive propositions:

(8.4.4) $\lambda p.\big(\big[\neg\sigma(0) \wedge (\forall q \forall r.(\sigma(q) \wedge \sigma(r)) \leftrightarrow \sigma(q \circ r))\big] \wedge \sigma(p)\big) \wedge abt(hesp, p)$

In the first conjunct of the above term, the first conjunct (in square brackets) describes the semantic value of σ as an information state. The rest of the term identifies the set of all propositions in σ which carry information about Hesperus.

The term from (8.4.5) generalizes the description from (8.4.4) across information states. In the term, ω is a variable over information states:

(8.4.5) $\lambda\omega\lambda p.\big(\big[\neg\omega(0) \wedge (\forall q \forall r.(\omega(q) \wedge \omega(r)) \leftrightarrow \omega(q \circ r))\big] \wedge \omega(p)\big) \wedge abt(hesp, p)$

It is easy to see that the resulting term is much more complex than the definition of the translation of Hesperus (or Bill) in the logic STY_1^3. To reduce the term's complexity, one could identify data-semantic propositions with functions from situations to truth-combinations, and identify information states with situations. However, the result of this simplification (in (8.4.6)) still has about the same complexity as the definition of the STY_1^3 translation of the name Bill (copied for convenience below). In fact, item (8.4.6) employs the *same* representational strategy as (8.4.1). This possibility has been demonstrated in Chapter 5.1.1 (cf. (5.1.3)).

(8.4.6) $\lambda i\lambda p.p(i) \wedge abt(hesp, p)$

(8.4.1) $\lambda i\lambda j\forall p.(p(i) \wedge abt(bill, p)) \twoheadrightarrow p(j)$

Our use of the type $(s\,s;t)$ as the basic type of the logic STY_1^3 is justified by the lower rank of the type $(s\,s;t)$ w.r.t. the rank of the type $(s\,(s;t);t)$ (cf. (8.4.6)) and by the analysis of natural language entailment as a truth- (rather than as an approximation-)ordering (cf. Def. 6.2.2). As a result of the former, only the type $(s\,s;t)$ satisfies the requirement of simplicity from Chapter 4. As a result of the latter, only the metatheory of our $(s\,s;t)$-based logic allows a restriction to the familiar logical connectives.

This completes our demonstration of the high complexity of name-interpretations in Data Semantics. We next show that the interpretations of names in semantic accounts of ellipsis have a similar complexity.

8.4.2. Interpretations of names in Nonsentential Speech. In (**Merchant, 2008**, Sect. 4), Merchant provides an analysis of sentential NP-interpretations which explains the semantic equivalence between proper names and sentences via an e-to-$((e\,s;t)\,s;t)$ meaning-shifting function. This function sends the traditional type-e referent, *partee*, of the name Barbara Partee to the result (in (8.4.7), next page) of applying its type-$((e\,s;t)\,s;t)$ correspondent, $\lambda P\lambda i.P(partee, i)$, to the type-$(e\,s;t)$ variable P_1. The term from (8.4.7) receives its semantic value via the contextually determined assignment function $g_{\mathcal{F}^2}$ (*ibid.* pp. 38–39). The inter-

pretation of this term in the designated TY_2^3 model $M_{\mathcal{F}^2}$ under the function $g_{\mathcal{F}^2}$ then yields the desired proposition.

$$\lambda i.P_1(partee, i) \tag{8.4.7}$$

Admittedly, Merchant's interpretation of the name Barbara Partee in a given context @ *looks* much simpler than the name's STY_1^3 interpretation at @. However, the result of replacing the variable P_1 by a contextually specified predicate (in (8.4.8)) already significantly increases the complexity of the term from (8.4.7):

$$\lambda i \forall P.P(partee, @) \twoheadrightarrow P(partee, i) \tag{8.4.8}$$

The generalization of the interpretation of Barbara Partee from (8.4.8) across contexts or situations yields the term from (8.4.9):

$$\lambda i \lambda j \forall P.P(partee, i) \twoheadrightarrow P(partee, j) \tag{8.4.9}$$

$$\lambda i \lambda j \forall p.\big(p(i) \wedge abt\,(bill, p)\big) \twoheadrightarrow p(j) \tag{8.4.1}$$

This term is equivalent to the definition of the STY_1^3 translation of Bill from (8.4.1) (again restated above). However, since our definition of the STY_1^3 translations of proper names facilitates the identification of relations between different individuals, and since it is closer in form to the $(s\,s;t)$-shifted variants of intensional formulas (cf. Def. 8.2.1), we choose this format.

We close this section with a discussion of the similarities between our strong single-type semantics and a propositional logic with only the Sheffer stroke. The discussion provides a different defense of the complexity of the TY_2^3 definitions of linguistic STY_1^3 translations.

8.4.3. Single-Type Objects and the Sheffer Stroke. The interpretation of the PTQ fragment through a single complex Montague type (s.t. individuals and propositions can both be represented via objects of this type) is reminiscent of the definition of the classical propositional connectives from only the negation of the conjunction operation (read 'not both', and called the *Sheffer stroke*, $|\,)$, cf. (**Sheffer, 1913**).[5] In particular, the Sheffer stroke enables the definition of $\neg\varphi$ as $(\varphi\,|\,\varphi)$[6], and enables the definitions of $(\varphi \to \psi)$, $(\varphi \wedge \psi)$, and $(\varphi \vee \psi)$ as $(\varphi\,|\,(\psi\,|\,\psi))$, $((\varphi\,|\,\psi)\,|\,(\varphi\,|\,\psi))$, and $((\varphi\,|\,\varphi)\,|\,(\psi\,|\,\psi))$, respectively, where φ and ψ are propositional constants. Since the semantics of the Sheffer stroke can be captured by a single logical axiom, cf. (**Nicod, 1917**), 'Sheffer stroke'-based systems improve upon the economy and simplicity of classical propositional logic without the loss of expressive power.

---

[5]The section originates from discussions with Leon Horsten, Jan Sprenger, and Dietmar Zaefferer.

[6]This holds because $\neg\varphi = \neg(\varphi \wedge \varphi)$.

Single-type semantics witnesses the merits of 'Sheffer-stroke' logics: In particular, it illustrates the greater simplicity (on the object level) and the higher degree of unification w.r.t. traditional Montague semantics (through the identification of new representational relations). However, the two systems share an important practical impediment. This impediment lies in the absence of a natural (i.e. linguistic or semantic/cognitive) correlate for the chosen primitive, and the resulting impossibility of using these systems *directly* for the translation or interpretation of natural language: For example, while most human languages command lexical translations of logical negation (e.g. not), conjunction (and, but), and disjunction (or), they lack a lexical correspondent of the Sheffer stroke. Similarly, while the explanation of a large number of linguistic facts (e.g. predication, anaphora, and wh-phenomena) require reference to individuals, few phenomena require reference to individual/proposition-neutral objects (cf. Ch. 1.2.1).

The salience of the connectives $\neg$, $\wedge$, and $\vee$ and of the types e and $(s;t)$ (or e, s, and t) is further reflected in our description of the Sheffer stroke in terms of $\neg$ and $\wedge$, and in our definition of single-type objects with reference to individuals, situations, and truth-values (cf. Ch. 5.3.1). But the salience of these 'natural' primitives can also be used to defend (and reduce) the complexity of the definitions of single-type objects: Once we have introduced the Sheffer stroke through reference to conjunction and negation, we do not require its re-introduction in the definition of the remaining connectives. In particular, it is redundant to define material implication, $(\varphi \rightarrow \psi)$, through the specification, $\neg(\varphi \wedge \neg(\psi \wedge \psi))$, of its 'Sheffer stroke'-definition.[7] A similar redundancy holds of the repeated specification of a term's TY_2^3 definition in our single-type semantics. In particular, it is unnecessary to describe each STY_1^3 interpretation of Bill through the relevant term from (C3). It suffices to know that – and how – the definition proceeds. Section 8.2.2 has demonstrated the latter. This insight motivates the observation from the subsequent paragraph of (8.2.3).

This completes our discussion of the semantic complexity of strong single-type objects. We close the chapter with a summary of our semantics' achievements. The main results of this dissertation are surveyed in the next Chapter (esp. in Sect. 9.1).

8.5. Summary

This chapter has developed a strong single-type semantics for Montague's PTQ fragment. The semantics is a designated model for the TY_2^3 subsystem STY_1^3 which interprets logical PTQ forms into constructions out of (equivalents of) propositional concepts (type $(s\,s;t)$). Since individuals and propositions can both be re-

[7] This holds because $\neg(\varphi \wedge \neg(\psi \wedge \psi)) = \neg(\varphi \wedge \neg\psi) = (\neg\varphi \vee \neg\neg\psi) = (\neg\varphi \vee \psi) = (\varphi \rightarrow \psi)$.

presented in the type $(s\,s;t)$, proper names, sentences, and complement phrases all receive an interpretation in this type (cf. Prop. 1.2). The resulting semantics preserves the merits of the 'pure' and the weak single-type semantics from Chapters 3 and 7. It improves upon these semantics by allowing for the possibility of assigning undefined truth-values to names (cf. Prop. 1.3.i) and by identifying a name's equivalent contextually salient sentences (cf. Prop. 1.3.ii). As a result of the latter, we describe the single-type semantics of the type $(s\,s;t)$ as a *strong* single-type semantics.

CHAPTER 9

Conclusion

This chapter evaluates the success of the experiment performed in this dissertation: *the attempt to support Partee's conjecture by constructing a single-type model for the PTQ fragment.* In particular, the chapter reviews the *conditions* which enable the formulation of a single-type semantics for the PTQ fragment, assesses the *strength* of these conditions on the proposed semantics, and *compares* these conditions with the conditions on other non-Montagovian semantics.

The chapter is organized as follows: To identify the challenges on the provision of a single-type semantics, Section 9.1 reviews the merits and limitations of our strong single-type semantics. Section 9.2 identifies the conditions on the solution of these challenges. Section 9.4 presents the main precursors of single-type semantics, and compares their conditions with the conditions on our strong single-type semantics. Section 9.3 answers the guiding questions from the introduction to this dissertation. Section 9.5 identifies alternative approaches to single-type semantics. The chapter closes with a view to future work.

9.1. Assessment of Single-Type Semantics

This dissertation has shown that Montague's PTQ fragment can be modeled through the use of a single basic type of object which is semantically neutral between individuals and propositions (Prop. 1.2). However, it has also shown (in Part I) that many semantic properties of basic-type objects (i.e. their truth-evaluability, and their equivalence to other non-algebraically related objects; Prop. 1.3) cannot be accommodated in a 'pure' single-type system which leaves the basic type unanalyzed. We have developed two alternative ('mixed') single-type semantics which identify the single basic type with a particular Montague type (in Part III), and have shown that these semantics satisfy the requirements from Proposition 1.3.

To evaluate the success of our single-type semantics, we consider the merits and limitations of the *strong* single-type semantics from Chapter 8. Our focus on this particular semantics is motivated by the fact that only strong single-type semantics model all of the observations from Proposition 1.3. As a result, we expect that these semantics will exemplify the advantages and drawbacks of single-type theories most clearly. We will see in the next section that – because of their good

suitability as formal semantics for the PTQ fragment – strong single-type semantics also best identify the constraints on any single-type semantics.

9.1.1. Merits of Single-Type Semantics. Chapter 1 has motivated our interest in single-type semantics with reference to its ability to identify new representational relations between different types of Montagovian objects (cf. Sect. 1.4.2), to its unificatory power (cf. Sect. 1.4.1), and to its extension of the modeling scope of traditional Montague semantics (cf. Sect. 1.2.1). Our formulation of the strong single-type semantics from Chapter 8 has confirmed these expectations. In particular, the injective functions from (9.1.1) and (9.1.2) (below) – which send Montagovian individuals and propositions to their strong single-type correlates – extend the set of the familiar type- (or meaning-)shifting functions from (**Partee, 1987**) (cf. Fig. 1.2):

$$\lambda x \lambda i \lambda j \forall p.\big(p(i) \wedge abt(x,p)\big) \twoheadrightarrow p(j) \tag{9.1.1}$$

$$\lambda p \lambda i \lambda j.p(j) \wedge \big(\forall q.\big(q(i) \wedge (\exists x.\, abt(x,q) \wedge abt(x,p))\big) \twoheadrightarrow q(j)\big) \tag{9.1.2}$$

Figure 9.1 (below) replaces the proxies, o_1 and o_2, for the single-type candidates from Chapters 7 and 8 by the types $(s;t)$ and $(s\,s;t)$. In the figure, the domains of the functions from (9.1.1) and (9.1.2) are connected by thick black arrows. Witnesses of the 'dashed' functions (e.g. for $(s;t)$-to-$(e\,s;t)$ shifts) are easily specified.

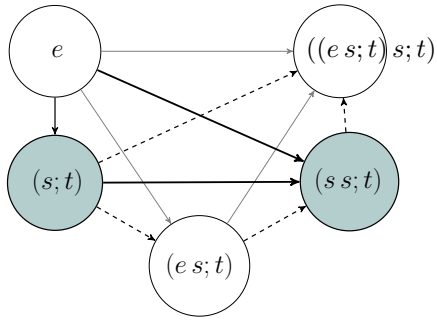

FIGURE 9.1. Relations between objects of the types from Figure 1.2 and objects of the type $(s;t)$ and $(s\,s;t)$.

The representation of Montagovian individuals and propositions by strong single-type objects (s.t. proper names, sentences, and complement phrases are all interpreted in the type $(s\,s;t)$) allows us to model a number of linguistic phenomena which cannot be accommodated in traditional Montague semantics. These phenomena include the neutrality of some verbs between an NP or a CP complement (cf. (8.2.20)), the possibility of coordinating proper names with complement phrases (cf. (8.2.23)), and the existence of specificational sentences with a CP comple-

ment (cf. (8.2.22)). Because of its 'rich' interpretation of logical PTQ forms (cf. (9.1.1), (9.1.2)), our strong single-type semantics also makes it possible to interpret isolated occurrences of proper names in the semantic type for sentences. The 'sentential' interpretation of names enables us to identify contextual truth- and equivalence-conditions for name/sentence pairs (cf. (8.3.5), (8.3.7)).

The different strategies for the representation of individuals and propositions from (9.1.1) and (9.1.2) further preserve the explanatory power of Montague semantics in our strong single-type semantics. This is due to the possibility of identifying two non-intersecting $(s\,s;t)$-subtypes whose elements have the form from (9.1.1) resp. (9.1.2). Since no element of either of these subtypes represents both an individual and a proposition, the hierarchy of type-$(s\,s;t)$ subtypes preserves all semantic distinctions of the ontology from (**Montague, 1970a**). As a result, our strong single-type semantics provides an equally 'good' explanation of syntactic well- (or ill-)formedness as Montague semantics.

The above observations identify our strong single-type semantics as a *monist conservative extension* of Montague semantics[1]. This description is justified by the semantics' assumption of a *single* basic type of object (s.t. it is a *monist* variant of Montague semantics), by its preservation of the explanatory power of Montague semantics (s.t. it is *conservative* over traditional Montague semantics), and by its ability to model a number of phenomena which cannot be modeled in traditional Montague semantics (s.t. our strong single-type semantics *extends* the empirical scope of this semantics).

9.1.2. Limitations of Single-Type Semantics. The success of our strong single-type semantics is curbed by the impossibility of achieving most[2] of the above-listed merits in the absence of a multi-typed metatheory. In particular, we have seen in Part I that a 'pure' single-type semantics is unable to serve as a foundation for syntactic categories (s.t. it is *not conservative* over Montague semantics), and that it is unable to identify equivalence relations between names and sentences (s.t. it does *not extend* the empirical scope of traditional Montague semantics to phenomena like (9)). As a result, a 'pure' single-type semantics is not, in all respects, preferable over traditional Montague semantics.

The merits of traditional Montague semantics and of our 'pure' and strong single-type semantics are compared in Table 9.1 (next page). In this table, the improvement (resp. regress) of a semantics w.r.t. the unification, and the explanatory and modeling power of its competing theories is marked by a checkmark, '✓' (resp. by a cross, '✗'). The abbreviations 'MS', 'TY_0', and 'STY_1^3' stand for '*Montague semantics*', '*'pure' single-type semantics*', and '*strong single-type semantics*':

[1]This description (as well as its extension from p. 193) has been proposed by Markus Werning.
[2]The only exception is the *unification* of Montague's semantic ontology.

Role of types	Better performed in MS	TY_0	STY^3_1
(1) *Unification of basic semantic domains* (monism)	✗	✓	✓
(2) *Explanation of distribution/categ's* (conservativity)	✓	✗	✓
(3) *Interpretation of natural language* (extension)	✗	✗	✓

Table 9.1. Success of single-typing vs. Montague typing.

However, we have seen in this dissertation that the limitations of 'pure' single-type semantics are not restricted to the inability of this semantics to achieve the merits from Section 9.1.1. Other challenges include the facts from (a) to (d):

(a) The truth-functional connectives are not available in the object theory.

(b) The truth of basic-type terms cannot be evaluated in the object theory.

(c) Equivalence between terms cannot be established in the object theory.

(d) Object-level separation subtypes cannot be identified within the single basic type.

These challenges will be addressed in the next section.

9.2. Constraints on Single-Type Semantics

The introduction to this chapter has already mentioned the dependence of a suitable single-type semantics on the satisfaction of a number of semantic conditions. For our strong single-type semantics, these conditions are listed below:

(i) The single basic type o is a *conjoinable* Montague type (cf. Ch. 4.1.1).

(ii) The metatheory commands the truth-value type t (cf. Ch. 2.5).

(iii) The metatheory commands types for individuals (e) and situations (s) (cf. Ch. 5.3.2).

(iv) Object-theoretic terms are defined by metatheoretic terms (cf. Ch. 5.3.1).

The relevance of each of these conditions for the solution to Challenges (a) to (d) from the end of the previous subsection is summarized in Table 9.2 (next page). In the table, conditions which are required for a solution to the respective challenge are marked by a checkmark. Conditions which are not required for the solution to the challenge are marked by a cross.

Note the strength of Conditions (i) to (iv) for the success of a single-type semantics: None of the challenges can be solved without the satisfaction of *at least two* conditions. Some important challenges (i.e. Challenge (c) and (d)) even require the satisfaction of *all* conditions. Since these conditions all refer to some sali-

Challenge \ Solution condition	(i)	(ii)	(iii)	(iv)
(a) *Definability of proxies for truth-funct. connectives*	✓	✓	✗	✗
(b) *Truth-evaluability of names/sentences* (Prop. 1.3.i)	✓	✓	✓	✗
(c) *Identifiability of name/S equivalences* (Prop. 1.3.ii)	✓	✓	✓	✓
(d) *Definability of single-type subsorts*	✓	✓	✓	✓

Table 9.2. Conditions on the success of a single-type semantics.

ent aspect of Montague's original type system, our single-type semantics remains *dependent* on Montague semantics.

The characterization of strong single-type semantics as a *monist dependent conservative extension* of Montague semantics challenges the status of single-type semantics as a serious competitor to Montague semantics. In particular, since our single-type semantics is unable to model the PTQ fragment without recourse to Montagovian (or Gallin) objects, it does not question the salience of Montague's original type system. To the contrary, it can be regarded as generating support for the semantic ontology from (**Montague, 1970a**; **1973**), cf. (**Gallin, 1975**).

Remarkably, the resistance of formal semantics to a 'pure' single-type characterization is not an isolated phenomenon. A similar observation has been made for the attempt to model natural language in an ontology based (only) on situations (cf. Sect. 9.4.1). Thus, in Barwise and Perry's interview on *Situations and Attitudes* (**Barwise and Perry, 1985**), the interviewer notes the following (p. 110):

> [...] in the early chapters of the book, you sketch a certain perspective: hard-headed realism [...]. The idea [is that] we should be able to understand [human language] in realistic terms without appeal to things like mental representations or other possible worlds. [...]
>
> But look what you end up with as parts of your reality: types of events, constraints, roles, frames of mind, ideas, concepts, and images [...]. You must admit that that sounds pretty far from the hard-headed realism of the early [chapters].

In Section 9.4, we will present other ostensible one-base-type semantics which are subject to similar constraints.

9.3. Answering the Initial Questions

The observations from the previous section enable us to formulate specific answers to the questions from the beginning of this thesis (cf. the first paragraph of Ch. 1.2). We present these answers in the original order of mention of their questions:

What happens if we replace Montague's types for individuals (e) *and propositions* ($\langle s,t\rangle$) *by a single basic type of object?*

ANSWER: Initially not much: We are still able to model all logical forms from the PTQ fragment. However, to ensure that basic-type forms have the expected semantic properties (like representability, truth, and equivalence), and to provide a semantic basis for some syntactic categories, we still need to refer to Montague types. As a result, a single-type semantics does not make Montague's type system dispensable.

What does a suitable interpretive domain for the single basic type look like?

ANSWER: To ensure the truth-evaluability of basic-type objects, the domain of the single basic type must be a construction to the type for propositions, $\langle s,t\rangle$. For the weak single-type semantics from Chapter 7, this requirement is satisfied by any type of the form $(\ldots\ s;\vec{t}\,)$, where $\vec{t}$ is a construction to the truth-value type t. For the strong single-type semantics from Chapter 8, this requirement is satisfied by any type of the form $(\ldots\ s\,s;\vec{t}\,)$. Objects of this type further enable the representation of individuals and propositions, and allow the interpretation of linguistic connectives as algebraic operations.

What effects does this change of type system have on our semantics' ability to model natural language?

ANSWER: The interpretation of proper names and sentences in a single basic type accommodates a number of phenomena which cannot be modeled in traditional Montague semantics (cf. Sect. 9.1.1). These phenomena include the neutrality of some verbs between an NP or a CP complement, the possibility of coordinating proper names with complement phrases, the existence of specificational sentences with a CP complement, and the local equivalence of names with sentences.

This completes our assessment of the strong single-type semantics from Chapter 8. We next present some principal precursors of this semantics, and discuss alternative strategies for the provision of a single-type semantics.

9.4. Precursors of Single-Type Semantics

This dissertation has treated single-type semantics as a novel theory of formal semantics. The attempt to support Partee's conjecture with a designated single-type theory is (to the best of our knowledge) unprecedented. However, the last three decades of semantic research have produced several *ostensible* one-base-type theories. These include Kratzer's Situation Semantics (**Kratzer, 1989**), Zalta's Abstract Object Theory (**Zalta, 1983**), and Chierchia and Turner's Property Theory (**Chierchia and Turner, 1988**). We discuss these theories in turn:

9.4.1. Situation Semantics. In (**Kratzer, 1989**), cf. (**Kratzer, 1990**; **2002**), Kratzer provides a formal semantics for counterfactual conditionals which is based on semantically primitive situations. Situations are parts of possible worlds (or *partial possible worlds*), cf. (**Barwise and Perry, 1983**), which contain possible individuals alongside possible facts. Situations are ordered by a 'part of'-relation, $\leq$, which captures the richness of their semantic content. The *truth* of a proposition p in a situation s is defined via the membership of s in p (s.t. p is true in s if $s \in p$). The *exemplification* of p by a fact s is defined via the p-minimality of s (s.t. s exemplifies p if s is the smallest situation in which p is true).

The association of possible individuals and facts with situations gives Kratzer's theory the appearance of a situation-based single-type semantics. However, her assignment of semantic types to syntactic categories destroys this appearance.[3] In particular, while Kratzer interprets individual constants and variables in the domain of situations (type s), cf. (**Kratzer, 1989**, p. 619), she identifies the semantic values of sentences with (functions from utterance situations to) *sets* of situations (type $(s;t)$, resp. $(s;(s;t))$) (*ibid.* pp. 615, 618). This move prevents the single-typed modeling of the PTQ fragment (cf. Prop. 1.2) and disables an accommodation of the phenomena from Chapter 1.2 (cf. Prop. 1.3).

Despite its unsuitability as a single-type semantics, Kratzer's semantics is closely related to the project of this dissertation. In particular, the weak single-type semantics from Part III contains a number of Kratzerian elements. These elements include the identification of *sets* of situations as a salient object type, and the use of a 'situations-to-sets of situations' lifting function. In (**Kratzer, 2002**, pp. 19–20), this function is described as a function from situations s to their smallest content-preserving extensions. Since possible individuals are classified as situations, this function also enables the lifting of individuals. Specifically, since Kratzer identifies every individual a with the smallest situation s in which this individual exists, the set denoted by $\{s' \mid s \leq s'\}$ is identical to the representation of individuals from (4.2.10).

We next discuss the relation of our single-type semantics to Zalta's Abstract Object Theory.

9.4.2. Abstract Object Theory. In (**Zalta, 1983**), cf. (**Zalta, 1988**), Zalta proposes an axiomatic metaphysics which seeks to describe the behavior of *abstract* objects. Abstract objects are non-concrete non-existing objects (e.g. Sherlock Holmes or Pegasus), which *cannot* have a location in space-time. They distinguish themselves from *ordinary* (i.e. concrete) objects (e.g. Ed or Barbara Partee), which *can* occupy a specific position in space-time. Objects of these two classes are

[3]However, since Kratzer's theory (like the theories from Sect. 9.4.2, 9.4.3) is not devised as a single-type semantics, it never intends or claims to satisfy Propositions 1.2 and/or 1.3.

associated with different ways of witnessing properties: While ordinary objects *exemplify* properties (e.g. biking to CSLI on Monday 4$^{\text{th}}$ November, 2013, at 8:51 a.m., resp. being a formal semanticist), abstract objects *encode* properties (e.g. being a consulting detective, resp. having two wings) (**Zalta, 1983**, pp. 12 ff.). Exemplification and encoding constitute a 'bottom up' and a 'top down' approach to predication, respectively (s.t. *a exemplifies P* at @ iff $\langle a, @\rangle \in P$, and *a encodes P* iff, for all w, $\langle P, w\rangle \in Q$, where a is a type-e correlate of Q).

Since abstract objects are individual correlates of certain sets of properties (s.t. there is an injective function from property sets to abstract objects, cf. (**Zalta, 1983**, p. 34)), abstract objects encode each of their encoded properties *by necessity*. For example, since Sherlock Holmes is a type-e correlate of the set including the properties of being a consulting detective and of living at 221b Baker Street, Sherlock Holmes encodes both of these properties. Since concrete objects do not serve as the individual correlates of property sets, they exemplify each of their properties *contingently*. For example, although Ed is a member of the set of cyclists at the actual world (or situation), it is well possible that he is a member of this set's complement at another world (or situation).

The possibility of encoding singleton sets of properties suggests the possibility of formulating single-type semantics in an 'abstract object'-framework. In particular, since every proposition φ can be converted into the property of being such that φ, there exists an 'abstract object'-correlate for each proposition, cf. (**Zalta, 1983**, p. 78). The union of the sets of ordinary and abstract objects then enables the construction of (a representation of) Montague's semantic ontology along the lines of Part I. By its proximity to our 'pure' single-type semantics, Zalta's theory accommodates NP/CP complement-neutral verbs, CP equatives, and NP/CP coordinations. However, since the theory does not specify entailment relations between ordinary and abstract objects, it cannot accommodate Proposition 1.3. As a result, the theory disqualifies as a single-type semantics for the PTQ fragment.

This completes our discussion of Abstract Object Theory. We finish our presentation of single-type precursors with a discussion of Chierchia and Turner's Property Theory.

9.4.3. Property Theory. In their effort to provide a (hyper-)intensional impredicative semantics for natural language predicative expressions, Chierchia and Turner (**1988**) adopt a single-typed semantics (their *Property Theory*, PT) which is not unlike the TY_0-based semantics from Part I.[4] Their theory is a variant of Henkin's Theory of Propositional Types, cf. (**Henkin, 1963**), which replaces the truth-value type t by a basic type, u, for basic individuals (called *urelements*).[5] Objects of this type take the form of familiar Montagovian individu-

[4]This similarity has been pointed out to me by Chris Potts.

als (our sort e) or of (hyper-)fine-grained propositions (called *information units*, sort i). The identity-conditions for information units remain unspecified.

Chierchia and Turners' linguistic application of Property Theory identifies this theory as a single-type semantics for a small extension of the PTQ fragment. Its interpretation of proper names and sentences as urelements, cf. (**Chierchia and Turner, 1988**, p. 284), supports Partee's conjecture from Proposition 1.2. However, since it (deliberately) restricts truth to information units (*ibid.* p. 270), PT excludes the truth-aptness of proper names (cf. Prop. 1.3.i). Since the connectives $\neg$, $\wedge$, $\vee$, $\forall$, $\exists$, $\rightarrow$, and $\leftrightarrow$ are also restricted to sort-i objects (*ibid.* p. 267), the theory is further unable to establish equivalence relations between names and sentences (cf. Prop. 1.3.ii). As a result, Property Theory also disqualifies as a good single-type semantics for the PTQ fragment.

The semantics from Part III answer the shortcomings of Property Theory as a single-type semantics. Our uniform treatment of proper names and sentences (s.t. names have truth-conditions, and can be coordinated via the designated single-type connectives; cf. Prop. 1.3.i) exemplifies this. Our definition of (the single-type correspondents of) individuals and information units in terms of their underlying individuals, respectively propositions further enables the specification of their equivalence-conditions (cf. Prop. 1.3.ii). Our definition of the designated single-type connectives in terms of the familiar propositional connectives follows this pattern. The development of Property Theory along the described lines remains a project for further research.

But the semantics from Part III also have a lesson to learn from Property Theory. This lesson concerns the organization of the base domain of our single-type logics into a *sorted* domain (analogous to the PT domain $D_u = (D_e \cup D_i)$). In Chapter 7.4, we have achieved this sorting of domains through the use of separation subtypes, cf. (**Lambek and Scott, 1986**; **Pollard, 2008**). Subtypes provide a rough semantic basis for syntactic categories, and reduce single-type domains to representationally relevant objects (along the lines of Ch. 4.1.2).

This completes our discussion of the precursors of single-type semantics. We next compare our adopted single-type strategy to other type-theoretic approaches to single-type semantics.[6]

9.5. Other Approaches to Single-Type Semantics

This dissertation has identified single-type models with models of a generalization of the simply typed lambda calculus, cf. (**Church, 1940**). These models give a *ty-*

[5]In fact, Chierchia and Turner replace the type t by the type e, which includes urelements and *nominalized functions*. Yet, since we are not interested in nominalization, we neglect the latter.
[6]The section originates from discussions with Stanley Peters, Daisuke Bekki, and Hans Leiss.

ped interpretation of the system from (**Church, 1985**), which involves the *explicit* and *unique typing* of every term (Fact 1). From the single basic type, all types are obtained through a generalization, (;), of the *function type* constructor $\rightarrow$ (cf. Montague's constructor $\langle\ ,\ \rangle$) (Fact 2).[7] Fact 1 ensures the well-typing of expressions and prevents functional self-application. Fact 2 identifies complex-type terms with the designators of function spaces.

The identification of single-type models with models of the simply typed lambda calculus follows the approach from (**Montague, 1973**), (**Gallin, 1975**), and (**Partee, 2006**). Our choice of this system is further supported by its generality and canonicity, cf. (**Scott, 1980**), and by the proximity of its models to models of set theory[8], cf. (**Henkin, 1950**). However, none of these facts excludes the possibility of providing a single-type semantics in models of other lambda calculi like the untyped lambda calculus, cf. (**Church, 1985**), or the polymorphically typed lambda calculus, cf. (**Girard, 1972**; **Reynolds, 1974**). These two systems flout Fact 2 and Fact 1, respectively. To motivate our choice of the single-type semantics from Part III, we briefly characterize an untyped and a polymorphically typed single-type semantics, and identify their merits and limitations.

9.5.1. Untyped Single-'Type' Semantics. The characterization of single-type semantics as a model of the *untyped lambda calculus* (**Church, 1985**), cf. (**Barendregt, 1984**), constitutes the conceptually simplest alternative to the semantics from Part III. The untyped lambda calculus is a *type-free* interpretation of the lambda calculus, in which all expressions remain untyped. As a result, we can identify the 'types' of the untyped lambda calculus with the *universal* type (our single basic type) into which all other types can be isomorphically embedded, cf. (**Scott, 1980**).

Like models of the untyped lambda calculus, models in an untyped single-'type' semantics can be described as functional domain structures, which consist of a complete lattice D, the space $(D \rightarrow D)$ of functions over elements of this lattice, a function sending every element of D to a corresponding function in the space $(D \rightarrow D)$, and the inverse of this function. To augment the language of the untyped lambda calculus with single-type stand-ins for logical constants, cf. (**Church, 1932**; **Henkin, 1950**), we demand that D be a complete *complemented* lattice, cf. (**Wadsworth, 1976**). We identify the top and bottom elements of this lattice with the semantic values of single-type terms of the form $\lambda \boldsymbol{x}_{\alpha}.\top$, resp. $\lambda \boldsymbol{x}_{\alpha}.\bot$, where α is the universal type. This move enables the interpretation of the single-type stand-ins for verum, falsum, etc. along the lines of Chapter 2.

[7]Since our type constructor (;) combines the constructors for *function types*, $\rightarrow$, and Cartesian *product types*, $\times$, we, in fact, already use a small extension of the simply typed lambda calculus.
[8]This holds by the existence of function spaces on sets, which associate with function types.

The assumption of a universal (single basic) type supports Partee's conjecture. The possibility of functional self-application further extends the scope of untyped single-type semantics to infinitives and gerunds (cf. the sentence Being fun is fun, cf. (**Chierchia and Turner, 1988**, p. 293)). However, since the untyped lambda calculus waives the distinction between functions and arguments, it fails to provide a formal basis for syntactic categories. As a result, it is not able to explain distributional phenomena in natural language. Since models of this calculus are moreover rather removed from the familiar Montagovian models, they are less suitable as a single-type semantics along the lines presented in Chapter 1.2.

9.5.2. Polymorphic Single-Type Semantics. The characterization of single-type semantics as a model of the *polymorphically typed lambda calculus*, cf. (**Girard, 1972**; **Reynolds, 1974**), preserves the merits of Montague's formal semantics. The polymorphically typed lambda calculus is an extension of the simply typed lambda calculus, cf. (**Church, 1940**), which allows the non-unique, i.e. polymorphic, typing of terms. As a result, a single term can denote objects of different types, and can be applied to differently typed arguments. A polymorphic single-type semantics will thus cover middle ground between the typed single-type semantics from Parts I and III, and the untyped single-type semantics from the previous subsection.

To obtain polymorphic types, we introduce a countable set of type variables on whose members we define a new form of universal abstraction, Λ. The abstraction operator Λ applies to terms and type variables to form a polymorphic function which takes types as arguments. The application of this function to a specific type yields the result of replacing all free occurrences of the type variable in the domain of the universal abstraction by that type.

The possibility of abstracting over a term's types enables us to replace the family of designated non-logical TY_0 constants $\{\doteq_{(\alpha\,\alpha;o)} \mid \alpha \in \mathsf{0Type}\}$ from Chapter 2.1 by the polymorphic constant $\doteq$. However, since the polymorphically typed lambda calculus has significantly different properties from the presented simply typed lambda calculus, its adoption as a framework for single-type semantics is not straightforwardly implemented. The formulation of a polymorphic single-type semantics is left as a future project.

9.6. Future Work

We close this thesis by identifying further projects for future work. These projects include the extension of our single-type semantics to the sentence-type interpretation of definite and indefinite noun phrases (Sect. 9.6.1), the unification of the semantic ontologies of larger fragments of natural language (Sect. 9.6.2), and the identification of the minimal number of types which are required for the interpretation of other linguistic fragments (Sect. 9.6.3).

9.6.1. Assertoric Interpretation of (In-)Definites. Our introduction to this dissertation has motivated Partee's conjecture with reference to the sentence-type interpretation of proper *names*. Notably, however, similar findings to the findings from (1) to (6) and (9) to (10) are also available for definite and indefinite noun phrases (hereafter, DPs). These findings include the neutrality of transitive verbs between a CP *and a DP* complement (cf. (11), (12)), the possibility of coordinating a CP *with a DP* (cf. (35), in App. B.2.2), and the sentential interpretation of *DP*s. The last-mentioned finding is illustrated in (25) and (26) (**Merchant, 2008**, pp. 10, 9), cf. (**Stainton, 2006**, p. 209):

(25) CONTEXT: Someone is about to take a seat on an empty chair. A person who is witnessing this act quickly points towards the chair and interjectingly utters (a).
   a. [$_{\text{NP}}$an editor of *Natural Language Semantics*]
   b. This seat is reserved for [$_{\text{NP}}$an editor of *Natural Language Semantics*].

(26) CONTEXT: Someone is trying to recognize a tune. Another person leans in on him and whistles (a).
   a. [$_{\text{NP}}$the song of mourning]
   b. The melody you are hearing is [$_{\text{NP}}$the song of mourning].

The provision of a basic-type interpretation of definite and indefinite noun phrases constitutes an easy extension of the presented single-type semantics: To accommodate these phenomena, we only need to extend the TY_2^3 language $\mathcal{L}^2$ via a (family of Skolemized) choice operator(s) (**Steedman, 2012**), cf. (**Winter, 2004; Kratzer, 1998**), introduce single-type proxies for these operators (for the basic-type translation of indefinite NPs), introduce a single-type proxy for the iota operator (for the translation of definite NPs), define the former through the latter (along the lines of Def. 8.2.2), and adapt our single-type translations of lexical items. The introduction of single-type proxies for the iota and choice operators is required by the fact that their types do not qualify as members of the single-type type system.

The resulting semantics will account for the truth-evaluability of definite and indefinite NPs (cf. Prop. 1.3.i) and for their semantic equivalence to sentences in a given context (cf. Prop. 1.3.ii).

9.6.2. Extension to Larger Fragments. Chapter 1 has limited the scope of this dissertation to the semantic ontology from (**Montague, 1970a**). As a result, our unificatory project has been confined to a neutralization of the distinction between individuals and propositions. It remains to investigate whether more recently introduced types (including the types for primitive propositions, situations, events, states, processes, registers, numbers, kinds, degrees, times, and intervals; cf. Ch. 1.1.3) admit of a similar single-type treatment.

For the case of situations, events, states, and processes, the answer is positive: Chapter 4.2.4 has already demonstrated the possibility of replacing possible worlds by situations in partial type theory. Kratzer (**2011**) has recently suggested the treatment of events as a particular subclass of situations. This treatment is made possible by the existence of 'minimal' situations, whose only true proposition is given by the affirmation of the event's culmination.[9] The lifting of this proposition to the type for propositional concepts (via the function from (9.1.2)), and the addition of an extra 'event-representation' argument to the STY^3_1 translations of verbs then allow the translation of the sentence Bill walks into the STY^3_1 term $\bigvee \boldsymbol{e}.\ \boldsymbol{walk}(\boldsymbol{e}, \boldsymbol{bill})$, where $\boldsymbol{e}$ is a variable over lifted events. The adaptation of the semantic constraints for the interpretation of STY^3_1 terms enables the expected definition of the above term (in (9.6.1)), where e is an event variable:

(9.6.1) $\lambda i \lambda j \exists e.\, walk(e, bill, j) \wedge \big(\forall p.(p(i) \wedge abt(bill, p)) \twoheadrightarrow p(j)\big)$

The extension of our strong single-type semantics to states and processes is analogously achieved, cf. (**Parsons, 1990**, Ch. 9).

The single-type treatment of registers, numbers, kinds, and degrees is suggested by the representability of these objects in the types $(s\, s; t)$ (for registers and numbers), $((s\, s; t)\, s\, s; t)$ (for kinds), and $(((s\, s; t)\, s\, s; t)\, s\, s; t)$ (for degrees), cf. (**Muskens, 1996**; **Chierchia, 2010**). The definition of the designator of their STY^3_1 representations remains a project for future research. Related projects include the formulation of a hyperintensional, (s, p)-based single-type semantics (with p the type for primitive propositions), and the investigation of the possibility of extending the presented theory to temporal intervals and times.

We close our suggestions for future work with a more abstract methodological project.

9.6.3. Reverse Formal Semantics. We have seen (in Ch. 5.3) that any suitable single-type semantics for the PTQ fragment relies on the assumption of a multi-typed metatheory with distinct types e, s, t.[10] We have also seen (in Ch. D.1; Ch. 7.2, 8.2) that puzzles involving intensionality can be solved in an $\{e, (s; t)\}$-based semantics, and that the possibility of defining the translations of the sentences from (1b) to (3b) and (7) requires an extension of the set of 'classical' Montagovian individuals with 'propositional' (i.e. abstract) individuals.

The above-mentioned insights constitute first steps towards the program of '*reverse formal semantics*'. This program – named in analogy with the corresponding program in the foundations of mathematics (cf. Ch. 5.2.2), cf. (**Friedman, 1975**; **Simpson, 2009**) – attempts to identify the *minimal* formal semantics (with

[9] In Kratzer's jargon: the existence of situations which *exemplify* the proposition associated with the event.

[10] This section is based on discussions with Cleo Condoravdi, Sam Sanders, and Seth Yalcin.

'minimality' defined w.r.t. the number and complexity of basic types) which interpret certain fragments of natural language. Questions in reverse formal semantics include the following:

(1) What *minimal number* of types is required for the interpretation of a certain fragment of natural language (e.g. the PTQ fragment w/o intensional nouns and verbs (1a), the PTQ fragment (1b), the extension of the PTQ fragment with NP/CP complement-neutral verbs (1c))?

(2) What is the *identity* of the types in the smallest sets of types from (1a) to (1c)? What are the objects in the types' domains?

(3) Which equinumerous type-sets are *equivalent* (up to coding) to the sets from (1) and (2) (s.t. they preserve the explanatory and modeling power of a semantics with basic types from the original set)?

(4) By (1) to (3): Which existing formal semantics (for which fragments?) can be *reduced* to which other semantics (for which other fragments)?

(5) Which types (if any) resist coding through the familiar types (s.t. they require the *jump* from one to another, non-equivalent set of types)?

This dissertation has provided an answer for particular cases of questions (1), (2), (4), and (5). In particular, it has shown that the interpretation of the PTQ fragment requires a minimum of *two* metalevel types (cf. (1b)). We have identified these types with the types for individuals (type e) and with functions from partial situations to truth-combinations (type $(s;t)$) (cf. (2b)). The possibility of avoiding Partee's temperature puzzle in this semantics suggests the possibility of reducing Gallin's formal semantics for the PTQ fragment (with types e, s, t) to (a variant) of the formal semantics from (**Montague, 1970a**) (cf. (4)). The impossibility of accommodating NP/CP complement-neutral verbs without an extension of Montague's type-e domain suggests the impossibility of reducing a semantics for (1c) to an $\{e, (s;t)\}$-based semantics (cf. (5)). A possible candidate for an equivalent of the set $\{e, (s;t)\}$ is the set $\{e, p\}$ from Section 9.6.2 (cf. (3)).

We hope that the answers to other cases of the above questions will identify new relations between existing theories of formal semantics, and that they will further our insight into the type system of natural language.

APPENDIX A

Abbreviations and Conventions

A.1. List of Abbreviations

ADV	verb phrase adverb
C	complementizer
CP	complement phrase
CT	Church's (**1940**) rule for the formation of (unary) function types
DET	determiner
DS	Deep Structure
IL	Intensional Logic
IT	Montague's (**1973**) rule for the formation of intensional types
IV	intransitive verb
LF	Logical Form
N	common noun
NP	noun phrase
P	preposition
PF	Phonological Form
PP	prepositional phrase
PT	Property Theory
PTQ	the paper (**Montague, 1973**)
QR	Quantifier Raising
S	sentence
SAV	sentence adverb
SCV	sentence-complement verb
SS	Surface Structure
ST	the single-type correlate of the rule **CT**
STT	Simple Theory of Types
STY_1^3	the 'strong' $(s\,s;t)$-based logic
TV	transitive verb
TY_0	the 'pure' single-type logic
TY_2	Gallin's (**1975**) type logic

TY_2^3	a partial n-ary functional variant of TY_2
VP	verb phrase
WTY_1^3	the 'weak' $(s;t)$-based logic

A.2. Notational Conventions

A.2.1. Non-Logical Constants.

Constant	TY_2^3 Type
john, *mary*, *bill*, *ninety*, *sherlock*, *moriarty*, *pat*, *partee*, c	e
@	s
ϕ	t
φ, ψ	$(s; t)$
π	$(s\, s; t)$
man, *woman*, *park*, *fish*, *pen*, *unicorn*, *problem*, *room*	$(e\, s; t)$
run, *walk*, *talk*, *wait*, *arrive*, E	$(e\, s; t)$
find, *lose*, *eat*, *love*, *date*, *remember*, *fear*, *destroy*, *hate*, *enter*	$(e\, e\, s; t)$
believe, *assert*	$((s; t)\, e\, s; t)$
temp, *price*, *rise*, *change*	$(((s; t); e)\, s; t)$
rapidly, *slowly*, *voluntary*, *allegedly*, *try*, *wish*	$((e\, s; t)\, e\, s; t)$
in, *for*	$(e\, (e\, s; t)\, e\, s; t)$
seek, *conceive*	$(((e\, s; t)\, s; t)\, e\, s; t)$
about	$(((e\, s; t)\, s; t)\, (e\, s; t)\, e\, s; t)$

Constant	TY_0 Type
$\boldsymbol{B}$, $\boldsymbol{C}$	$(\alpha_1 \ldots \alpha_n; o)$
$\lceil$	$((\alpha_1 \ldots \alpha_n; o)\, \alpha_1 \ldots \alpha_n; o)$
$\barwedge$, $\barvee$	$((\alpha_1 \ldots \alpha_n; o)\, (\alpha_1 \ldots \alpha_n; o)\, \alpha_1 \ldots \alpha_n; o)$
$\bigwedge, \bigvee$	$(\alpha\, o; o)$
$\dot{\Rightarrow}$, $\dot{=}$, $\dot{\neq}$, $\dot{\rightarrow}$, $\dot{\leftrightarrow}$	$(\alpha\, \alpha; o)$
$\top, \bot$, ***john***, ***mary***, ***bill***, ***ninety***, ***sherlock***, ***moriarty***, ***pat***, ***partee***, $\boldsymbol{w}$	o
$\boxdot, \diamondsuit$, ***man***, ***woman***, ***park***, ***fish***, ***pen***, ***unicorn***, ***room***, ***problem***	$(o; o)$
run, ***walk***, ***talk***, ***wait***, ***arrive***, $\boldsymbol{E}$	$(o; o)$
find, ***lose***, ***eat***, ***love***, ***date***, ***remember***, ***fear***, ***destroy***, ***hate***, ***enter***, ***believe***, ***assert***	$(o\, o; o)$
temp, ***price***, ***rise***, ***change***	$((o; o); o)$
rapidly, ***slowly***, ***voluntary***, ***allegedly***, ***try***, ***wish***	$((o; o)\, o; o)$
in, ***for***	$(o\, (o; o)\, o; o)$
seek, ***conceive***	$(((o; o); o)\, o; o)$
about	$(((o; o); o)\, (o; o)\, o; o)$

A.2.2. Variables.

Variable	TY_2^3 Type	Object
$i, j, k, k_1, \ldots, k_n$	s	situation
$x, x_1, \ldots, x_n, y, z$	e	individual
$p, p_1, \ldots, p_n, q, r$	$(s; t)$	proposition
$P, P_1, \ldots, P_n$	$(e\, s; t)$	1st-order intensional indiv. p'ty
$T, T_1, \ldots, T_n$	$((s; t); e)$	proposition-to-individual-fct.
$Q, Q_1, \ldots, Q_n$	$((e\, s; t)\, s; t)$	2nd-order intensional indiv. p'ty
$L, L_1, \ldots, L_n$	$((((e\, s; t)\, s; t)\, s; t)\, s; t)$	4th-order intensional indiv. p'ty

Variable	TY_0 Type	Object
$\boldsymbol{x}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n, \boldsymbol{y}, \boldsymbol{z}$	o	entity
$\boldsymbol{p}, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_n, \boldsymbol{q}, \boldsymbol{r}$	o	entity
$\boldsymbol{P}, \boldsymbol{P}_1, \ldots, \boldsymbol{P}_n$	$(o; o)$	1st-order property of entities
$\boldsymbol{Q}, \boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_n$	$((o; o); o)$	2nd-order property of entities
$\boldsymbol{L}, \boldsymbol{L}_1, \ldots, \boldsymbol{L}_n$	$(((o; o); o)\, o; o)$	

A.3. Glossary

conjoinable type	a type of the form $(\alpha_1 \dots \alpha_n; t)$
entity	an object of the type o; a member of the set $\mathcal{O}$
extension	an object which is not an *intension*
formula	a term of the type t
index	an object of the type s; a *situation* in the set W
individual	an object of the type e; a member of the set $\mathcal{A}$
intension	a type-$(\alpha_1 \dots \alpha_{n-1}\, s; t)$ or -$(s; (\alpha_1 \dots \alpha_{n-1}; t))$ object
intensional formula	a term of the type $(s; t)$
property	an object of some type $(\alpha_1 \dots \alpha_n; t)$, where $2 \leq n \in \mathbb{N}$ if $\alpha_n = s$ and $n \geq 1$ otherwise
proposition	an object of the type $(s; t)$
propositional type	a type of the form $(\alpha_1 \dots \alpha_{n-1}\, s; t)$
(possible) situation	a partial *index* w s.t., for some prop. p, $w \notin p$ and $w \notin \neg p$
propositional concept	an object of the type $(s; (s; t))$ (or $(s\, s; t)$)
truth-combination	an object of the type t; here, a member of the set **3**
truth-value	a classical type-t object; a member of the set **2**
truth-fct'l connectives	the logical constants $*, \bot, \Rightarrow, =, \neg, \wedge$, etc.
single-type connectives	the non-logical constants $\textcircled{1}, \dot{\Rightarrow}, \doteq, \vdash, \barwedge$, etc.
(possible) world	a total *index* w s.t., for all prop's p, $w \in p$ or $w \in \neg p$

APPENDIX B

Further Motivation for Partee's Conjecture

This appendix surveys the original motivation for Partee's conjecture, cf. (**Partee, 2006**, pp. 37–38), (in Sect. B.1) and extends the empirical motivation for a single-type semantics from Section 1.2.1 (in Sect. B.2). The original motivation for Partee's conjecture lies in the existence of support for a syntactic analogue of Proposition 1.2 (i.e. for Carstairs-McCarthy's *single-category hypothesis*). The new empirical motivation for Partee's conjecture lies in the existence of further empirical support for single-type semantics, and in a more detailed explanation of some of the examples from Chapter 1.2.1.

B.1. Partee's Original Motivation

In his monograph *The Origins of Complex Language* (**1999**), Andrew Carstairs-McCarthy makes the following claim about natural language syntax:

PROPOSITION B.1 (Single-Category Hypothesis). *The syntactic distinction between noun phrases and sentences is inessential for the generation of complex modern languages. Their grammatical category system can be constructed from <u>one</u> basic category.*

To acknowledge its original proponent, we will sometimes refer to Proposition B.1 as *Carstairs-McCarthy's conjecture*. This conjecture suggests the possibility of obtaining all grammatical categories from a single basic category (dubbed 'X'), which is neutral between the syntactic categories for noun phrases (NP), sentences (S), and complement phrases (CP). From the category X, all other categories are obtained via the rule **CC**:

(**CC**) *If A and B are syntactic categories, then C (whose members are generated by the phrase structure rule $C \longrightarrow A$, $A \longrightarrow BC$, or $A \longrightarrow CB$) is a category.*

Since the new category X is neutral between the categories NP and S, non-basic categories will display less distinctions than the familiar non-basic categories from categorial syntax: For example, since complementizers (C) and sentence adverbs (SAV) both merge with an expression of the category X to form an expression of the category X, Carstairs-McCarthy's single-category syntax will also neutralize

the distinction between these two categories. The same holds of the distinction between transitive verbs (e.g. find) and sentence-complement verbs (believe).

Carstairs-McCarthy supports his hypothesis by citing four classes of support, which are taken from evolutionary linguistics, the syntax of nominalization, language acquisition, and pragmatics. We discuss each of them in turn:

B.1.1. Support from Syntactic Complexity. Carstairs-McCarthy's principal motivation for Proposition B.1 concerns the evolutionary contingency of the distinction between sentences and noun phrases, and the resulting possibility of an equally suitable single-category syntax. In particular, his argument for Proposition B.1 identifies the 'sentence/NP'-distinction as a byproduct of neural mechanisms which are associated with changes in the human vocal tract (**Carstairs-McCarthy, 1999**, pp. 121–129):

At the end of the Pliocene, the descent of his larynx enabled Homo erectus for the first time to produce 'longer' (i.e. temporally extended) sounds. According to Carstairs-McCarthy, this could have been achieved either through an expansion of the existing vocabulary, or through the introduction of a speech-patterning mechanism. Since the internal structure of syllables constituted an *existing* instance of the second alternative, its transfer to the new 'higher' linguistic level was the simplest way of increasing the language's expressive power. More complex expressions were then obtained by forming multi-word lexical strings in which noun phrases and sentences mimicked the function of onsets and syllables, respectively:

(27) a. $[\mathrm{k}_{\mathrm{onset}}[\mathrm{i}\!:_{\mathrm{nucleus}}]_{\mathrm{rhyme}}]_{\mathrm{syllable}}$ :: $[_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{Bill}][_{\mathrm{VP}}[_{\mathrm{IV}}\mathsf{walks}]]]$

b. $[\mathrm{k}_{\mathrm{onset}}[\mathrm{æ}_{\mathrm{nucleus}}\ \mathrm{t}_{\mathrm{coda}}]_{\mathrm{rhyme}}]_{\mathrm{syllable}}$ :: $[_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{Bill}][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{loves}][_{\mathrm{NP}}\mathsf{Mary}]]]$

Like syllables constitute the smallest unit of organization for a sequence of speech sounds, sentences constitute the smallest unit of organization for a sequence of words. Like rhymes combine with an onset to form a syllable, verb phrases combine with a noun phrase to form a sentence.

Carstairs-McCarthy motivates the contingency of the 'sentence/noun phrase'-distinction with reference to their development in analogy with syllable structure. Note, however, that Carstairs-McCarthy's evidence for Proposition B.1 does not refute mainstream theories of syntax. Rather, it supports the equal suitability of an alternative, single-category syntax for natural language. The possibility of such a syntax is further supported by the easy possibility of converting sentences into noun phrases (Sect. B.1.2), and by the ambiguity between an NP's referential and assertoric interpretation (Sect. B.1.3, B.1.4).

B.1.2. Evidence from Nominalization. To support the possibility of an expressive single-category language, Carstairs-McCarthy assumes the hypothetical language *Nominalized English*, whose syntax contains only one basic category,

NP. From this class of expressions, he obtains all familiar syntactic categories – except for the category 'sentence' – as derived categories. Sentences like (28) are converted into noun phrases (e.g. (28a)–(28c)) via a simple nominalization procedure (**Carstairs-McCarthy, 1999**, p. 23).

(28) John saw a snake.
  a. the snake that John saw
  b. a snake that John saw
  c. John, who saw a snake

Since noun phrases like (28a) to (28c) exhibit assertoric force, intransitive verbs (walk) and sentence adverbs (e.g. necessarily) qualify as members of the same syntactic category.

B.1.3. Evidence from Language Acquisition. Carstairs-McCarthy further supports his hypothesis by showing that the ambiguity between an NP's referential and assertoric reading (cf. (28a)–(28c), resp. (28)) is not idiosyncratic to Nominalized English, but constitutes a widespread linguistic phenomenon. To this aim, he cites recent research into primate communication systems (**Cheney and Seyfarth, 1990**). This research has found that the language of Vervet monkeys – which commands only a single-class lexicon – displays a significant illocutionary variance. Thus, the one-word locution eagle from (29) can be interpreted as any of the expressions from (29a)–(29d) (**Carstairs-McCarthy, 1999**, pp. 21–22, 46):

(29) Eagle
  a. an eagle (referential)
  b. There's an eagle over there. (assertoric)
  c. The eagle constitutes a threat. (assertoric)
  d. Hide from the eagle!/Take cover! (imperative)

B.1.4. Evidence from Pragmatics. Carstairs-McCarthy completes his argument by noting that the 'referential/assertoric' ambiguity of noun phrases is not restricted to the developmental domain, but constitutes a salient feature of spoken adult language. To illustrate the assertoric function of noun phrases, he lists a number of expressions (in (30)–(32)) which make the same assertion as their sentential glosses (in square brackets) (**Carstairs-McCarthy, 2005**, p. 151):

(30) The arrival of the Queen of Sheba [The Queen of Sheba arrives]
(31) Victory for Dewey [Dewey wins]
(32) Lord and Lady Blenkinsop [Here are Lord and Lady Blenkinsop]

Carstairs-McCarthy supports the assertoric interpretation of items (30) to (32) by showing that they have truth- (or satisfiability-)conditions, and that they can be

regarded as true or false in a given situation, cf. (**Carstairs-McCarthy, 2005**, p. 151). Thus, the use of (30) as a description of the arrival of the *First Harlot* or of the *departure* of the Queen of Sheba is not only infelicitous, but false. The premature newspaper headline from (31) (when Dewey has, in fact, lost the presidential elections) asserts a similar falsity.

The above suggests the conceptual possibility of a single-category language with the expressive power of our familiar natural languages.

Carstairs-McCarthy's arguments from Sections B.1.2 and B.1.4 are directly transferable to natural language semantics. In particular, his examples from (30) to (32) complement Merchant's motivation for Proposition 1.3 (cf. Ch. 1.2.1). Examples (28) and (28a) to (28c) support the semantic similarity between sentences and indefinite NPs, cf. (**Partee, 2006**, p. 38). Under the (naïve) assumption of a close relation between syntactic categories and semantic types, the support for Proposition B.1 can thus be taken as a motivation for Proposition 1.2.

B.2. New Motivation for Partee's Conjecture

The introduction to this dissertation has motivated Partee's conjecture by considerations from lexical syntax, coordination, specification, and nonsentential speech. The present section adds further examples of some of these phenomena and provides a more detailed explanation of some earlier examples.

B.2.1. More Motivation from Lexical Syntax. Examples (1) to (3) have demonstrated the ability of single-type semantics to accommodate NP/CP complement-neutral verbs. This subsection identifies another NP/CP neutral form (i.e. prepositions) and shows that our semantics can accommodate its neutrality.

Since single-type semantics interprets proper names and complement phrases in the same semantic type, it accommodates the ambiguity of German prepositional phrases between the results of combining a preposition (e.g. vor, durch) with an NP, and of combining a pro-preposition (e.g. *da*vor, *da*durch) with a CP.[1] The first possibility is witnessed by the phrases vor Moriarty [Engl.: 'of Moriarty'] and durch einen Pfeiler [Engl.: 'through a beam'] in (33a), resp. (34a). The second possibility is witnessed by the phrases *da*vor, dass Moriarty ihn zerstört [Engl.: '*there*-of that Moriarty him destroys', i.e. 'of Moriarty destroying him'] and *da*durch, dass er einen Pfeiler aufstellt [Engl.: '*there*-through that he a beam puts up', i.e. 'by putting up a beam'] in (33b), resp. (34b).[2] Below, 'PRO-' and 'GC' are the labels for pro-elements and for gerundial clauses, respectively. German-language examples are supplied with a literal English translation and an informal gloss:

(33) a. Sherlock hat Angst vor [$_{\text{NP}}$Moriarty].
[*literal:* Sherlock has fear of [$_{\text{NP}}$Moriarty].]
[*gloss:* Sherlock is afraid of [$_{\text{NP}}$Moriarty].]

b. Sherlock hat Angst *da*vor, [$_{\text{CP}}$dass Moriarty ihn zerstört].
[*literal:* Sherlock has fear PRO-of [$_{\text{CP}}$that Moriarty him destroys].]
[*gloss:* Sherlock is afraid of [$_{\text{GC}}$Moriarty destroying him].]

(34) a. Peter stützt das Dach durch [$_{\text{NP}}$einen Pfeiler].
[*literal:* Peter supports the roof through [$_{\text{NP}}$a beam].]

b. Peter stützt das Dach *da*durch, [$_{\text{CP}}$dass er einen Pfeiler aufstellt].
[*literal:* Peter supports the roof PRO-through [$_{\text{CP}}$that he a beam puts up].]
[*gloss:* Peter is supporting the roof by [$_{\text{GC}}$putting up a beam].]

[1] I owe this observation to Markus Werning.

[2] A similar phenomenon is observed in Dutch: cf. the sentences Sherlock is bang voor [$_{\text{NP}}$Moriarty] (cf. (33a)) and Sherlock is *er*voor bang [$_{\text{CP}}$dat Moriarty hem vernietigt] (cf. (33b)), respectively Pieter steunt het dak door (middel van) [$_{\text{NP}}$een balk] (cf. (34a)) and Pieter steunt het dak [$_{\text{CP}}$doordat hij een balk installeert] (cf. (34b)).

Since Montague semantics interprets prepositions as expressions of the type $\langle \underline{e}, \langle \langle e, \langle s,t\rangle\rangle, \langle e, \langle s,t\rangle\rangle\rangle\rangle$, it succeeds in modeling (33a) and (34a). However, the sentences (33b) and (34b) defy accommodation in traditional Montague semantics. This is due to the fact that the pronominal character of *da-* prevents its interpretation as a meaning-shifter of prepositional arguments (type-$\langle \underline{e}, \langle \langle e, \langle s,t\rangle\rangle, \langle e, \langle s,t\rangle\rangle\rangle\rangle$-to-$\langle \langle \underline{s,t}\rangle, \langle \langle e, \langle s,t\rangle\rangle, \langle e, \langle s,t\rangle\rangle\rangle\rangle$). However, only this interpretation enables the combination of the pro-prepositions davor and dadurch with a CP.

Since the first argument place of our new type of prepositions, $\langle \underline{o}, \langle \langle o,o\rangle, \langle o,o\rangle\rangle\rangle$, is neutral between names and complement phrases, single-type semantics can model both members of the pairs of sentences from (33) and (34).

The next section presents additional support for the semantic equivalence of noun phrases and complement phrases (cf. (9)–(12)).

B.2.2. More Support for Name/Sentence Equivalences. Our investigation of the interpretations of (9a) and (10a) in the contexts from (9) and (10) has found that referential noun phrases can be related to complement phrases by (mutual) semantic entailment. The obtaining of these relations is supported by the intuitive redundancy of some NP/CP-coordinations, and by the intuitive contradiction that results from denying (the sentence associated with) one of their conjuncts.

Consider the NP/CP coordination from (5) (copied, for convenience, below) and the results (in (35), (36)) of coordinating the complements of the different occurrences of the verb notice from (11) and of coordinating the different prepositional phrases from (34):

(5) Sherlock fears [$_{\text{NP}}$Moriarty] *and* [$_{\text{CP}}$that Moriarty will destroy him].

(35) Chris noticed [$_{\text{NP}}$the problem] *and* [$_{\text{CP}}$that Mary hates Bill].

(36) Peter stützt das Dach durch [$_{\text{NP}}$einen Pfeiler] *und da*durch, [$_{\text{CP}}$dass er einen Pfeiler aufstellt].

[*gloss:* Peter is supporting the roof through [$_{\text{NP}}$a beam] *and* by [$_{\text{GC}}$putting up a beam].]

All three of the above sentences engender a feeling of redundancy. This feeling is brought about by the fact that the CP in the second conjunct (in (5), the phrase that Moriarty will destroy him) contains the information of (the sentential interpretation of) the NP in the first conjunct (here, of the name Moriarty), such that the CP semantically *entails* (the sentential interpretation of) the NP. Since, in (35) and (36), the CP further does not contain any information which is not already encoded in (the sentential interpretation of) the relevant NP (s.t. the NP also semantically entails the CP), the NP and CP from (35) and (36) are, in fact, semantically *equivalent*.[3]

For the above sentences, the obtaining of (mutual) semantic entailment relations between the CP and the sententially interpreted NP is supported by the possibility of replacing the common coordinating conjunctions and and und in (5), (35), and (36) by the specifying conjunction viz. [Ger.: 'nämlich'], cf. (**Kraak and Klooster, 1968**; **Koster, 2000**). The resulting sentences (in (37)–(39)) express that the complement of the second occurrence of the NP/CP neutral expression specifies the complement of the first occurrence of the NP/CP neutral expression:

(37) Sherlock fears [$_{\text{NP}}$Moriarty], *viz.* [$_{\text{CP}}$that Moriarty will destroy him].

(38) Chris noticed [$_{\text{NP}}$the problem], *viz.* [$_{\text{CP}}$that Mary hates Bill].

(39) Peter stützt das Dach durch [$_{\text{NP}}$einen Pfeiler]; *nämlich* dadurch, [$_{\text{CP}}$dass er einen Pfeiler aufstellt].
[*gloss:* Peter is supporting the roof through [$_{\text{NP}}$a beam], *viz.* by [$_{\text{GC}}$putting up a beam].]

The obtaining of entailment relations between sentences (or complement phrases) and the sentential interpretations of referential NPs is further supported by the difficulty of negating an NP's sentential interpretation in the conjunct of a coordinated NP/CP-structure, cf. (**Elugardo and Stainton, 2005**, p. 5). Specifically, once we have learned that Sherlock fears being destroyed by Moriarty, learning that he does, however, not fear Moriarty[4] yields incompatible information (cf. (40)). The coordination of the phrases from (11b) and (34b) with the results of negating the content of the phrases from (11a) and (34a) (in (41a), resp. (42a)), and the coordination of the phrases from (11a) and (34a) with the results of negating the content of the phrases from (41b) and (34b) yield similar results. The contradictoriness of the coordinations from (41b) and (42b) again supports the obtaining of a *mutual* entailment relation between the relevant NP and CP. In (40) to (42), negations are written in script font:

(40) * Sherlock DOESN'T fear [Moriarty], *but* [$_{\text{CP}}$that Moriarty will destroy him].

(41) a. * Chris FAILED TO notice [$_{\text{NP}}$the problem], *but* noticed [$_{\text{CP}}$that Mary hates Bill].

b. * Chris noticed [$_{\text{NP}}$the problem], *but* FAILED TO notice [$_{\text{CP}}$that Mary hates Bill].

(42) a. * Peter stützt das Dach NICHT durch [$_{\text{NP}}$einen Pfeiler], *sondern* dadurch, [$_{\text{CP}}$dass er einen Pfeiler aufstellt].

[3]Note that, in contrast to the identification of the equivalences from (9) and (10), the identification of the equivalences from (35) and (36) does not require a specification of the relevant situational context.

[4]Since the negation of proper names is typically not available in English, we negate instead the first occurrence of the verb fears.

[*gloss:* * Peter is NOT supporting the roof through [$_{GC}$a beam], *but* by [$_{GC}$putting up a beam].]

b. * Peter stützt das Dach durch [$_{NP}$einen Pfeiler], *aber* NICHT dadurch, [$_{CP}$dass er einen Pfeiler aufstellt].

[*gloss:* * Peter is supporting the roof through [$_{GC}$a beam], *but* NOT by [$_{GC}$putting up a beam].]

This completes our further empirical motivation of the desirability of a single-type semantics.

APPENDIX C

Proofs

C.1. Derived Sequent Rules for TY_0

On the basis of the sequent rules from Tables 2.1 and 2.2, and Notation 2.1.1, we derive the rules for the TY_0 stand-ins for *verum*, inequality, disjunction, implication, and biimplication, together with the De Morgan laws (DM_1, DM_2), the rules of double negation (DNI, DNE), and the rule of contraposition (CP) as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, \textcircled{\scriptsize$\bot$} \Rightarrow \Delta}\,\textcircled{\scriptsize$\bot$}\mathsf{L}}{\Gamma, \textcircled{\scriptsize$\bot$} \Rightarrow \Delta, \textcircled{\scriptsize$\bot$}}\,W\mathsf{R}}{\Gamma \Rightarrow \Delta, (\textcircled{\scriptsize$\bot$} \doteq \textcircled{\scriptsize$\bot$})}\,\doteq\mathsf{R}}{\Gamma \Rightarrow \Delta, \textcircled{\scriptsize$\top$}}\ \text{def. } \textcircled{\scriptsize$\top$}\ \ [\textcircled{\scriptsize$\top$}\mathsf{R}]$$

$$\dfrac{\dfrac{\dfrac{}{\Gamma \Rightarrow \Delta, \boldsymbol{A} \doteq \boldsymbol{A}}\,\doteq\mathsf{R}}{\Gamma, \neg \boldsymbol{A} \doteq \boldsymbol{A} \Rightarrow \Delta}\,\neg\mathsf{L}}{\Gamma, \boldsymbol{A} \not\doteq \boldsymbol{A} \Rightarrow \Delta}\ \text{def. } \not\doteq\ [\not\doteq\mathsf{L}]$$

$$\dfrac{\dfrac{\dfrac{\Gamma, \boldsymbol{A} \Rightarrow \Delta}{\Gamma, \neg\Delta \Rightarrow \neg\boldsymbol{A}}\,\neg\mathsf{R} \qquad \dfrac{\Gamma, \boldsymbol{B} \Rightarrow \Delta}{\Gamma, \neg\Delta \Rightarrow \neg\boldsymbol{B}}\,\neg\mathsf{R}}{\Gamma, \neg\Delta \Rightarrow (\neg\boldsymbol{A} \barwedge \neg\boldsymbol{B})}\,\barwedge\mathsf{R}}{\Gamma, \neg(\neg\boldsymbol{A} \barwedge \neg\boldsymbol{B}) \Rightarrow \Delta}\,\neg\mathsf{L}}{\Gamma, (\boldsymbol{A} \barvee \boldsymbol{B}) \Rightarrow \Delta}\ \text{def. } \barvee\ [\barvee\mathsf{L}]$$

$$\dfrac{\dfrac{\dfrac{\Gamma \Rightarrow \Delta, \boldsymbol{A}, \boldsymbol{B}}{\neg\boldsymbol{A}, \neg\boldsymbol{B} \Rightarrow \Delta, \neg\Gamma}\,\neg\mathsf{R}}{(\neg\boldsymbol{A} \barwedge \neg\boldsymbol{B}) \Rightarrow \Delta, \neg\Gamma}\,\barwedge\mathsf{L}}{\Gamma \Rightarrow \Delta, \neg(\neg\boldsymbol{A} \barwedge \neg\boldsymbol{B})}\,\neg\mathsf{R}}{\Gamma \Rightarrow \Delta, (\boldsymbol{A} \barvee \boldsymbol{B})}\ \text{def. } \barvee\ [\barvee\mathsf{R}]$$

$$\dfrac{\dfrac{\dfrac{\Gamma \Rightarrow \Delta, \boldsymbol{A}}{\Gamma, \neg\boldsymbol{A} \Rightarrow \Delta}\,\neg\mathsf{L} \qquad \Gamma, \boldsymbol{B} \Rightarrow \Delta}{\Gamma, (\neg\boldsymbol{A} \barvee \boldsymbol{B}) \Rightarrow \Delta}\,\barvee\mathsf{L}}{\Gamma, (\boldsymbol{A} \bar{\to} \boldsymbol{B}) \Rightarrow \Delta}\ \text{def. } \bar{\to}\ [\bar{\to}\mathsf{L}]$$

$$\dfrac{\dfrac{\dfrac{\Gamma, \boldsymbol{A} \Rightarrow \Delta, \boldsymbol{B}}{\Gamma \Rightarrow \Delta, \neg\boldsymbol{A}, \boldsymbol{B}}\,\neg\mathsf{R}}{\Gamma \Rightarrow \Delta, (\neg\boldsymbol{A} \barvee \boldsymbol{B})}\,\barvee\mathsf{R}}{\Gamma \Rightarrow \Delta, (\boldsymbol{A} \bar{\to} \boldsymbol{B})}\ \text{def. } \bar{\to}\ [\bar{\to}\mathsf{R}]$$

$$\dfrac{\dfrac{\dfrac{\Gamma \Rightarrow \Delta, \boldsymbol{A}, \boldsymbol{B} \qquad \Gamma, \boldsymbol{A} \Rightarrow \Delta, \boldsymbol{A}}{\Gamma, (\boldsymbol{B} \dot{\rightarrow} \boldsymbol{A}) \Rightarrow \Delta, \boldsymbol{A}}\,\dot{\rightarrow}\mathsf{L} \qquad \dfrac{\Gamma, \boldsymbol{B} \Rightarrow \Delta, \boldsymbol{B} \qquad \Gamma, \boldsymbol{A}, \boldsymbol{B} \Rightarrow \Delta}{\Gamma, \boldsymbol{B}, (\boldsymbol{B} \dot{\rightarrow} \boldsymbol{A}) \Rightarrow \Delta}\,\dot{\rightarrow}\mathsf{L}}{\Gamma, (\boldsymbol{A} \dot{\rightarrow} \boldsymbol{B}), (\boldsymbol{B} \dot{\rightarrow} \boldsymbol{A}) \Rightarrow \Delta}\,\dot{\rightarrow}\mathsf{L}}{\dfrac{\Gamma, ((\boldsymbol{A} \dot{\rightarrow} \boldsymbol{B}) \mathbin{\dot{\wedge}} (\boldsymbol{B} \dot{\rightarrow} \boldsymbol{A})) \Rightarrow \Delta}{\Gamma, (\boldsymbol{A} \dot{\leftrightarrow} \boldsymbol{B}) \Rightarrow \Delta}\,\text{def. } \dot{\leftrightarrow}\ [\,\dot{\leftrightarrow}\mathsf{L}]}\,\dot{\wedge}\mathsf{L}$$

$$\dfrac{\dfrac{\dfrac{\Gamma, \boldsymbol{A} \Rightarrow \Delta, \boldsymbol{B}}{\Gamma \Rightarrow \Delta, (\boldsymbol{A} \dot{\rightarrow} \boldsymbol{B})}\,\dot{\rightarrow}\mathsf{R} \qquad \dfrac{\Gamma, \boldsymbol{B} \Rightarrow \Delta, \boldsymbol{A}}{\Gamma \Rightarrow \Delta, (\boldsymbol{B} \dot{\rightarrow} \boldsymbol{A})}\,\dot{\rightarrow}\mathsf{R}}{\Gamma \Rightarrow \Delta, ((\boldsymbol{A} \dot{\rightarrow} \boldsymbol{B}) \mathbin{\dot{\wedge}} (\boldsymbol{B} \dot{\rightarrow} \boldsymbol{A}))}\,\dot{\wedge}\mathsf{R}}{\Gamma \Rightarrow \Delta, (\boldsymbol{A} \dot{\leftrightarrow} \boldsymbol{B})}\,\text{def. } \dot{\leftrightarrow}\ [\,\dot{\leftrightarrow}\mathsf{R}]$$

$$\dfrac{\dfrac{\dfrac{\Gamma \Rightarrow \Delta, (\neg \boldsymbol{A} \mathbin{\dot{\vee}} \neg \boldsymbol{B})}{\boldsymbol{A}, \boldsymbol{B} \Rightarrow \Delta, \neg\Gamma}\,\neg\mathsf{R}}{(\boldsymbol{A} \mathbin{\dot{\wedge}} \boldsymbol{B}) \Rightarrow \Delta, \neg\Gamma}\,\dot{\wedge}\mathsf{L}}{\Gamma \Rightarrow \Delta, \neg(\boldsymbol{A} \wedge \boldsymbol{B})}\,\neg\mathsf{R}\ [\mathsf{DM}_1] \qquad \dfrac{\dfrac{\dfrac{\Gamma \Rightarrow \Delta, \neg \boldsymbol{A}}{\boldsymbol{A} \Rightarrow \Delta, \neg\Gamma}\,\neg\mathsf{R} \qquad \dfrac{\Gamma \Rightarrow \Delta, \neg \boldsymbol{B}}{\boldsymbol{B} \Rightarrow \Delta, \neg\Gamma}\,\neg\mathsf{R}}{(\boldsymbol{A} \mathbin{\dot{\vee}} \boldsymbol{B}) \Rightarrow \Delta, \neg\Gamma}\,\dot{\vee}\mathsf{L}}{\Gamma \Rightarrow \Delta, \neg(\boldsymbol{A} \mathbin{\dot{\vee}} \boldsymbol{B})}\,\neg\mathsf{R}\ [\mathsf{DM}_2]$$

$$\dfrac{\neg \boldsymbol{A} \Rightarrow \neg \boldsymbol{A}}{\boldsymbol{A} \Rightarrow \neg\neg \boldsymbol{A}}\,\neg\mathsf{R}\ [\mathsf{DNI}] \qquad \dfrac{\neg \boldsymbol{A} \Rightarrow \neg \boldsymbol{A}}{\neg\neg \boldsymbol{A} \Rightarrow \boldsymbol{A}}\,\neg\mathsf{L}\ [\mathsf{DNE}]$$

$$\dfrac{\dfrac{\Delta \Rightarrow \Gamma}{\Delta \Rightarrow \neg\neg\Gamma}\,\mathsf{DNI}}{\neg\Gamma \Rightarrow \neg\Delta}\,\neg\mathsf{R}\ [\mathsf{CP}]$$

C.2. Proofs of Theorems

Theorem 2.3 (Soundness)**.** *For all sets* Γ, Δ *of* TY_0 *terms, if* $\Gamma \vdash_{\mathrm{TY}_0} \Delta$*, then* $\Gamma \models_g \Delta$.

PROOF. Assume that $\Gamma \vdash_{\mathrm{TY}_0} \Delta$. We prove by induction on the height of the proof that every line $\Gamma \Rightarrow \Delta$ is valid. For reasons of space, we limit ourselves to the most interesting case, $\bigwedge \mathsf{R}$.

Assume that $\Gamma \Rightarrow \Delta$ (with $\Delta := \{\Sigma, \bigwedge \boldsymbol{x}.\, \boldsymbol{A}\}$) is a conclusion of $\bigwedge \mathsf{R}$, such that $\Gamma \not\models_g \Delta$. Since $\Gamma \Rightarrow \Delta$ is the first 'bad' line in the proof, it follows by the induction hypothesis that $\Gamma \models_g \Sigma, \boldsymbol{A}\{\boldsymbol{x} := \boldsymbol{c}\}$ in a TY_0 model $M_F = \langle F, I_F, V_F\rangle$ under the assignment g_F, where $\boldsymbol{c}$ is fresh. Since $M_F \not\models_g \Sigma, \bigwedge \boldsymbol{x}.\, \boldsymbol{A}$, there exists some $d \in F$ such that $M_F \not\models_g \Sigma, \boldsymbol{A}\{\boldsymbol{x} := d\}$. Let V'_F be like V_F with the difference that $V'_F(\boldsymbol{c}) = d$, and let $M'_F = \langle F, I_F, V'_F\rangle$. Then, $M'_F \not\models_g \boldsymbol{A}\{\boldsymbol{x} := \boldsymbol{c}\}$ and, since $\boldsymbol{c}$ does not occur in $\Gamma \cup \{\boldsymbol{A}\}$, $M'_F \not\models_g \Sigma, \bigwedge \boldsymbol{x}.\, \boldsymbol{A}$. But this violates the induction hypothesis. □

Theorem 2.4 (Model existence)**.** *Let* L *and* $\mathcal{C}$ *be* TY_0 *languages s.t.* $L \cap \mathcal{C} = \emptyset$*, where every set* $\mathcal{C}_\alpha$ *is countably infinite. Assume that* $\mathfrak{P}$ *is a sound provability property w.r.t.* $L \cup \mathcal{C}$*, and that* Π *is a sequent in* L*. If* $\Pi \notin \mathfrak{P}$*, then* Π *is refutable by a countable* TY_0 *model.*

Our proof of Theorem 2.4 closely follows the one in (**Muskens, 1999**; **2007**). There, it is first shown that sequents which result from an unsuccessful attempt at constructing a Gentzen-style proof from the bottom up (called *Hintikka sequents*) are refutable. This fact is then used to establish the refutability of a large class of sequents.

To facilitate the definition of Hintikka sequents, we represent sequents as pairs $\{\mathsf{L}\colon \boldsymbol{A}, \mathsf{R}\colon \boldsymbol{A}\}$ of signed TY_0 terms, where the signs L (for 'left') and R ('right') indicate the terms' structural position in the sequent. For Γ and Δ as above, the sequent $\Gamma \Rightarrow \Delta$ is then represented by the set $\{\mathsf{L}\colon \boldsymbol{A} \mid \boldsymbol{A} \in \Gamma\} \cup \{\mathsf{R}\colon \boldsymbol{A} \mid \boldsymbol{A} \in \Delta\}$.

We hereafter abbreviate '$\Gamma \Rightarrow \Delta$' as '$\Pi$'. Hintikka sequents are then defined as follows:

DEFINITION C.2.1 (Hintikka sequents). A sequent Π of the logic TY_0 is a *Hintikka sequent* if one of the following holds:

(i) $\{\mathsf{L}\colon \boldsymbol{A}, \mathsf{R}\colon \boldsymbol{A}\} \not\subseteq \Pi$ if $\boldsymbol{A} \in T_o$;
(ii) $\mathsf{L}\colon ⓘ \notin \Pi$;
(iii) $\mathsf{L}\colon (\lambda \boldsymbol{x}.\, \boldsymbol{A})(\boldsymbol{B})(\vec{\boldsymbol{C}}) \in \Pi \Longrightarrow \mathsf{L}\colon \boldsymbol{A}\{\boldsymbol{x} := \boldsymbol{B}\}(\vec{\boldsymbol{C}}) \in \Pi$ for all closed $(\lambda \boldsymbol{x}.\, \boldsymbol{A})$, $\boldsymbol{B}$, $\vec{\boldsymbol{C}}$ of appropriate type, where $\vec{\boldsymbol{C}}$ is a sequence of TY_0 terms;
(iv) $\mathsf{R}\colon (\lambda \boldsymbol{x}.\, \boldsymbol{A})(\boldsymbol{B})(\vec{\boldsymbol{C}}) \in \Pi \Longrightarrow \mathsf{R}\colon \boldsymbol{A}\{\boldsymbol{x} := \boldsymbol{B}\}(\vec{\boldsymbol{C}}) \in \Pi$ for all closed $(\lambda \boldsymbol{x}.\, \boldsymbol{A})$, $\boldsymbol{B}$, and $\vec{\boldsymbol{C}}$ of appropriate type;
(v) $\mathsf{L}\colon (\boldsymbol{A} \Rightarrow \boldsymbol{B}) \in \Pi \Longrightarrow \mathsf{L}\colon \boldsymbol{B}(\vec{\boldsymbol{C}}) \in \Pi$ or $\boldsymbol{A}(\vec{\boldsymbol{C}}) \in \Pi$ for all closed $\boldsymbol{A}$, $\boldsymbol{B}$,

and $\vec{\boldsymbol{C}}$ of appropriate type;

(vi) $\mathsf{L}\colon(\boldsymbol{A} \Rightarrow \boldsymbol{B}) \in \Pi \Longrightarrow$ there are constants $\vec{\boldsymbol{c}}$ of appropriate type s.t. $\{\mathsf{L}\colon \boldsymbol{A}\,(\vec{\boldsymbol{c}}), \mathsf{R}\colon \boldsymbol{B}\,(\vec{\boldsymbol{c}})\} \subseteq \Pi$.

We call a Hintikka sequent Π *complete* if $\mathsf{L}\colon \boldsymbol{A} \in \Pi$ or $\mathsf{R}\colon \boldsymbol{A} \in \Pi$ for every $\boldsymbol{A} \in L$.

We next establish the refutability of Hintikka sequents by countable TY_0 models:

LEMMA 1 (Hintikka). *Every Hintikka sequent* Π *is refutable by a* TY_0 *model. If* Π *is complete, it is refutable by a countable* TY_0 *model.*

PROOF. Via the construction of a TY_0 model M_F refuting Π. By elementary considerations, M_F is countable if Π is complete. We identify the interpretation of TY_0 terms with their equivalence classes under equality. Below, we further assume that $\circledast$ is a nullary TY_0 constant for the *undefined* entity. By induction on the number of connectives in a TY_0 term, we then establish that, for every $\boldsymbol{A}$,

(a) $\mathsf{L}\colon \quad \boldsymbol{A} \in \Pi \Longrightarrow M_F \models_g \boldsymbol{A}$;
(b) $\mathsf{R}\colon \quad \boldsymbol{A} \in \Pi \Longrightarrow M_F \models_g \ulcorner \boldsymbol{A}, \circledast$;
(c) $\mathsf{L}\colon \ulcorner \boldsymbol{A} \in \Pi \Longrightarrow M_F \models_g \ulcorner \boldsymbol{A}$;
(d) $\mathsf{R}\colon \ulcorner \boldsymbol{A} \in \Pi \Longrightarrow M_F \models_g \boldsymbol{A}, \circledast$.

It follows that the model M_F refutes the Hintikka sequent Π. □

Our proof of the model existence theorem for TY_0 is based on the notion of a provability property. The latter has the following definition:

DEFINITION C.2.2 (Provability property). Let $\mathfrak{P}$ be a set of sequents in the language L. The property $\mathfrak{P}$ is a *provability property with respect to* L if $\mathfrak{P}$ is closed under the sequent rules such that, if $\{\Pi_1, \ldots, \Pi_n\} \subseteq \mathfrak{P}$ and if $\Pi_1, \ldots, \Pi_n \backslash \Pi$ is a sequent rule, then $\Pi \in \mathfrak{P}$.

A provability property in L is sound if no $\Pi \in \mathfrak{P}$ is refuted by a TY_0 model for L.

Theorem 2.4 establishes that sequents which are not members of a sound provability property in an extended language can be extended to Hintikka sequents in that language, and are, thus, refutable.

Theorem 2.4 (Model existence)**.** *Let* L *and* $\mathcal{C}$ *be* TY_0 *languages s.t.* $L \cap \mathcal{C} = \emptyset$*, where every set* $\mathcal{C}_\alpha$ *is countably infinite. Assume that* $\mathfrak{P}$ *is a sound provability property with respect to* $L \cup \mathcal{C}$*, and that* Π *is a sequent in* L*. If* $\Pi \notin \mathfrak{P}$*, then* Π *is refutable by a countable* TY_0 *model.*

PROOF. Via the construction of a Hintikka sequent Π^* such that $\Pi \subseteq \Pi^*$. Let $\vartheta_1, \ldots, \vartheta_n, \ldots$ be an enumeration of all signed sentences in $L \cup \mathcal{C}$, and let $\iota(\vartheta)$ denote the index which the signed sentence ϑ obtains in this enumeration. For every natural number n, we define a sequent Π by the following induction:

Let $\Pi_0 = \Pi$. We define Π_{n+1} as follows:

$$\Pi_{n+1} = \begin{cases} \Pi_n & \text{if } \Pi_n \cup \{\vartheta_n\} \in \mathfrak{P}; \\ \Pi_n \cup \{\vartheta_n\} & \text{if } \Pi_n \cup \{\vartheta_n\} \notin \mathfrak{P} \text{ and } \vartheta_n \text{ is not of the form } \mathsf{R}\colon \bigwedge \boldsymbol{x}_\alpha.\, \boldsymbol{A} \text{ or } \mathsf{L}\colon \bigwedge \boldsymbol{x}_\alpha.\neg \boldsymbol{A}; \\ \Pi_n \cup \{\vartheta_n, \mathsf{R}\colon \boldsymbol{A}\{\boldsymbol{x} := \boldsymbol{c}\}, \mathsf{L}\colon \neg \boldsymbol{A}\{\boldsymbol{x} := \boldsymbol{c}\}\} & \text{if } \Pi_n \cup \{\vartheta_n\} \notin \mathfrak{P} \text{ and } \vartheta_n \text{ is of the form } \mathsf{R}\colon \bigwedge \boldsymbol{x}_\alpha.\, \boldsymbol{A} \text{ or } \mathsf{L}\colon \bigwedge \boldsymbol{x}_\alpha.\neg \boldsymbol{A}, \text{ where } \boldsymbol{c} \text{ is the first constant in } \mathcal{C}_\alpha \text{ that does not occur in } \Pi_n \cup \{\vartheta_n\}. \end{cases}$$

That $\Pi_n \notin \mathfrak{P}$ for every n follows by a simple induction which uses Definition C.2.2, together with the fact that $\bigwedge \mathsf{R}$ is a sequent rule for TY_0.

Define $\Pi^* = \cup_n \Pi_n$. We next establish that, for all finite sets $\{\vartheta_{k_1}, \ldots, \vartheta_{k_n}\}$ and for all $k \geq max\{k_1, \ldots, k_n\}$, the following holds:

$$\{\vartheta_{k_1}, \ldots, \vartheta_{k_n}\} \subseteq \Pi^* \quad \Leftrightarrow \quad \Pi_k \cup \{\vartheta_{k_1}, \ldots, \vartheta_{k_n}\} \notin \mathfrak{P}\,. \tag{C.2.1}$$

We then verify through the use of (C.2.1) that Π^* is a Hintikka sequent. Consequently, Π^* is refutable by a general model for the logic TY_0. We show that Π^* (and, hence, Π) is refutable by a countable TY_0 model via a proof that Π^* is complete. □

C.3. Definitions of Designated WTY_1^3 and STY_1^3 Constants

This section derives the definitions of the 'weak' and the 'strong' single-type stand-ins for the remaining truth-functional connectives and quantifiers. To do this, we use the definitions of the designated single-type constants $\textcircled{\scriptsize\bot}$ and $\dot{\Rightarrow}$ (cf. (C1), (C2)), and the definitions of the TY_0 stand-ins from Notation 2.1.1.

We first derive the TY_2^3 definitions of the 'weak' single-type constants from Notation 7.2.1:

(C.3.1)
$$\begin{aligned}
&\big(\bigwedge \boldsymbol{x}.\boldsymbol{A}\big)\\
=\;&\big((\lambda\boldsymbol{x}.\textcircled{\scriptsize\top}) \mathrel{\dot{\Rightarrow}} (\lambda\boldsymbol{x}.\boldsymbol{A})\big)\\
=\;&\big((\lambda\boldsymbol{x}\lambda j.\top) \mathrel{\dot{\Rightarrow}} (\lambda\boldsymbol{x}\lambda k.\boldsymbol{A}(k))\big)\\
=\;&\big(\lambda i.(\lambda\boldsymbol{x}\lambda j.\top)(i) \Rightarrow (\lambda\boldsymbol{x}\lambda k.\boldsymbol{A}(k))(i)\big)\\
=\;&\big(\lambda i.(\lambda\boldsymbol{x}.\top) \Rightarrow (\lambda\boldsymbol{x}.\boldsymbol{A}(i))\big)\\
=\;&\big(\lambda i\forall\boldsymbol{x}.\boldsymbol{A}(i)\big)
\end{aligned}$$

(C.3.2)
$$\begin{aligned}
&\boldsymbol{B} \doteq \boldsymbol{C}\\
=\;&\big(\bigwedge \boldsymbol{Y}.\boldsymbol{Y}(\boldsymbol{B}) \mathrel{\dot{\Rightarrow}} \boldsymbol{Y}(\boldsymbol{C})\big)\\
=\;&\big(\lambda i\forall\boldsymbol{Y}.[\lambda j.\,\boldsymbol{Y}(\boldsymbol{B})(j) \Rightarrow \boldsymbol{Y}(\boldsymbol{C})(j)](i)\big)\\
=\;&\big(\lambda i\forall\boldsymbol{Y}.[\boldsymbol{Y}(\boldsymbol{B})(i) \Rightarrow \boldsymbol{Y}(\boldsymbol{C})(i)]\big)\\
=\;&\big(\lambda\boldsymbol{x}\lambda i.\boldsymbol{B}(\boldsymbol{x},i) = \boldsymbol{C}(\boldsymbol{x},i)\big)
\end{aligned}$$

(C.3.3)
$$\begin{aligned}
&\neg\boldsymbol{B}\\
=\;&\big(\lambda\boldsymbol{x}.\boldsymbol{B}(\boldsymbol{x}) \doteq \textcircled{\scriptsize\bot}\big)\\
=\;&\big(\lambda\boldsymbol{x}\lambda i.\boldsymbol{B}(\boldsymbol{x},i) = [\lambda j.\bot](i)\big)\\
=\;&\big(\lambda\boldsymbol{x}\lambda i.\boldsymbol{B}(\boldsymbol{x},i) = \bot\big)\\
=\;&\big(\lambda\boldsymbol{x}\lambda i.\neg\boldsymbol{B}(\boldsymbol{x},i)\big)
\end{aligned}$$

(C.3.4)
$$\begin{aligned}
&(\boldsymbol{B} \wedge \boldsymbol{C})\\
=\;&\big(\lambda\boldsymbol{x}.(\lambda\boldsymbol{X}.\boldsymbol{X}(\boldsymbol{B} \doteq \boldsymbol{C})) \doteq (\lambda\boldsymbol{X}.\boldsymbol{X}(\textcircled{\scriptsize\top}))\big)\\
=\;&\big(\lambda\boldsymbol{x}\lambda i.(\lambda\boldsymbol{X}.\boldsymbol{X}(\lambda j.\boldsymbol{B}(j) = \boldsymbol{C}(j)))(i) = (\lambda\boldsymbol{X}.\boldsymbol{X}(\lambda k.\top))(i)\big)\\
=\;&\big(\lambda\boldsymbol{x}\lambda i.\boldsymbol{B}(\boldsymbol{x},i) \wedge \boldsymbol{C}(\boldsymbol{x},i)\big)
\end{aligned}$$

The TY_2^3 definitions of the 'strong' single-type constants from Notation 8.2.1 are derived below:

(C.3.5)
$$\begin{aligned}
&\textcircled{\scriptsize T}\\
=\ &(\textcircled{\scriptsize $\perp$} \doteq\!\Rightarrow \textcircled{\scriptsize $\perp$})\\
=\ &(\lambda i\lambda j.(\lambda k\lambda k_1.\perp)(i,j) \Rightarrow (\lambda k\lambda k_1.\perp)(i,j))\\
=\ &(\lambda i\lambda j.\perp \Rightarrow \perp)\\
=\ &(\lambda i\lambda j.\top)
\end{aligned}$$

(C.3.6)
$$\begin{aligned}
&\big(\textstyle\bigwedge \boldsymbol{x}.\boldsymbol{A}\big)\\
=\ &\big((\lambda \boldsymbol{x}.\textcircled{\scriptsize T}) \doteq\!\Rightarrow (\lambda \boldsymbol{x}.\boldsymbol{A})\big)\\
=\ &\big((\lambda \boldsymbol{x}\lambda k\lambda k_1.\top) \doteq\!\Rightarrow (\lambda \boldsymbol{x}\lambda k_2\lambda k_3.\boldsymbol{A}(k_2,k_3))\big)\\
=\ &\big(\lambda i\lambda j.(\lambda \boldsymbol{x}\lambda k\lambda k_1.\top)(i,j) \Rightarrow (\lambda \boldsymbol{x}\lambda k_2\lambda k_3.\boldsymbol{A}(k_2,k_3))(i,j)\big)\\
=\ &\big(\lambda i\lambda j.(\lambda \boldsymbol{x}.\top) \Rightarrow (\lambda \boldsymbol{x}.\boldsymbol{A}(i,j))\big)\\
=\ &\big(\lambda i\lambda j\forall \boldsymbol{x}.\boldsymbol{A}(i,j)\big)
\end{aligned}$$

(C.3.7)
$$\begin{aligned}
&\boldsymbol{B} \doteq \boldsymbol{C}\\
=\ &\big(\textstyle\bigwedge \boldsymbol{Y}.\boldsymbol{Y}(\boldsymbol{B}) \doteq\!\Rightarrow \boldsymbol{Y}(\boldsymbol{C})\big)\\
=\ &\big(\lambda i\lambda j\forall \boldsymbol{Y}.[\lambda k\lambda k_1.\boldsymbol{Y}(\boldsymbol{B})(k,k_1) \Rightarrow \boldsymbol{Y}(\boldsymbol{C})(k,k_1)](i,j)\big)\\
=\ &\big(\lambda i\lambda j\forall \boldsymbol{Y}.[\boldsymbol{Y}(\boldsymbol{B})(i,j) \Rightarrow \boldsymbol{Y}(\boldsymbol{C})(i,j)]\big)\\
=\ &\big(\lambda \boldsymbol{x}\lambda i\lambda j.\boldsymbol{B}(\boldsymbol{x},i,j) = \boldsymbol{C}(\boldsymbol{x},i,j)\big)
\end{aligned}$$

(C.3.8)
$$\begin{aligned}
&\neg\boldsymbol{B}\\
=\ &\big(\lambda \boldsymbol{x}.\boldsymbol{B}(\boldsymbol{x}) \doteq \textcircled{\scriptsize $\perp$}\big)\\
=\ &\big(\lambda \boldsymbol{x}\lambda i\lambda j.\boldsymbol{B}(\boldsymbol{x},i,j) = [\lambda k\lambda k_1.\perp](i,j)\big)\\
=\ &\big(\lambda \boldsymbol{x}\lambda i\lambda j.\boldsymbol{B}(\boldsymbol{x},i,j) = \perp\big)\\
=\ &\big(\lambda \boldsymbol{x}\lambda i\lambda j.\neg\boldsymbol{B}(\boldsymbol{x},i,j)\big)
\end{aligned}$$

(C.3.9)
$$\begin{aligned}
&(\boldsymbol{B} \wedge \boldsymbol{C})\\
=\ &\big(\lambda \boldsymbol{x}.(\lambda \boldsymbol{X}.\boldsymbol{X}(\boldsymbol{B} \doteq \boldsymbol{C})) \doteq (\lambda \boldsymbol{X}.\boldsymbol{X}(\textcircled{\scriptsize T}))\big)\\
=\ &\big(\lambda \boldsymbol{x}\lambda i\lambda j.(\lambda \boldsymbol{X}.\boldsymbol{X}(\lambda k\lambda k_1.\boldsymbol{B}(j) = \boldsymbol{C}(k,k_1)))(i,j) =\\
&\qquad\qquad (\lambda \boldsymbol{X}.\boldsymbol{X}(\lambda k_2\lambda k_3.\top))(i,j)\big)\\
=\ &\big(\lambda \boldsymbol{x}\lambda i\lambda j.\boldsymbol{B}(\boldsymbol{x},i,j) \wedge \boldsymbol{C}(\boldsymbol{x},i,j)\big)
\end{aligned}$$

(C.3.10)
$$\begin{aligned}
& \boxdot \boldsymbol{A} \\
= \; & \big(\bigwedge \boldsymbol{p}.\boldsymbol{p} \doteq \textcircled{\perp} \vee (\boldsymbol{p} \doteRightarrow \boldsymbol{A})\big) \\
= \; & \big(\bigwedge \boldsymbol{p}.(\lambda k_2 \lambda k_3.\boldsymbol{p}\,(k_2,k_3) = (\lambda k_6 \lambda k_7.\perp)(k_2,k_3)) \vee \\
& \qquad (\lambda k_4 \lambda k_5.\boldsymbol{p}\,(k_4,k_5) \Rightarrow \boldsymbol{A}(k_4,k_5))\big) \\
= \; & \big(\bigwedge \boldsymbol{p}\big[\lambda k \lambda k_1.(\lambda k_2 \lambda k_3.\boldsymbol{p}\,(k_2,k_3) = \perp)(k,k_1) \vee \\
& \qquad (\lambda k_4,k_5.\boldsymbol{p}\,(k_4,k_5) \Rightarrow \boldsymbol{A}(k_4,k_5))(k,k_1)\big]\big) \\
= \; & \big(\lambda i \lambda j \forall \boldsymbol{p}.[\lambda k \lambda k_1.\boldsymbol{p}\,(k,k_1) = \perp \vee (\boldsymbol{p}\,(k,k_1) \Rightarrow \boldsymbol{A}(k,k_1))](i,j)\big) \\
= \; & \big(\lambda i \lambda j \forall \boldsymbol{p}.\boldsymbol{p}\,(i,j) = \perp \vee (\boldsymbol{p}\,(i,j) \Rightarrow \boldsymbol{A}(i,j))\big) \\
= \; & \big(\lambda i \lambda j \forall \boldsymbol{p}.\boldsymbol{p}(i,j) \Rightarrow \boldsymbol{A}\,(i,j)\big) \\
= \; & (\lambda i \lambda j \forall \boldsymbol{p}.\boldsymbol{p}\,(i,j) = [\exists w_s \forall q.q(w) \twoheadrightarrow q(i)] \Rightarrow \boldsymbol{A}(i,j)) \\
= \; & (\lambda i \lambda j \forall k.\boldsymbol{A}(i,k)) \\
= \; & (\lambda i \lambda j.\Box \boldsymbol{A}(i))
\end{aligned}$$

The equivalence of $\big(\lambda i \forall q.q(@) \twoheadrightarrow q(i)\big)^\bullet$ (cf. (C0)) with the TY_2^3 term from (8.2.2) is shown in (C.3.11):

(C.3.11)
$$\begin{aligned}
& \boldsymbol{w} \\
= \; & \big(\lambda i \forall q.q(@) \twoheadrightarrow q(i)\big)^\bullet \\
= \; & \lambda i \lambda j.\underline{[\forall q.q(@) \twoheadrightarrow q(j)]} \wedge \\
& \qquad \big(\forall p.\big(p(i) \wedge \big(\exists x.\big(abt\,\big(x,\underline{[\lambda k \forall q.q(@) \twoheadrightarrow q(k)]}\big) \wedge abt\,(x,p)\big)\big)\big) \twoheadrightarrow p(j)\big) \\
= \; & \lambda i \lambda j.\underline{[\forall q.q(@) \twoheadrightarrow q(j)]} \wedge \big(\forall p.\big(p(i) \wedge \big(\exists x.\big(abt\,\big(x,\underline{\big[\lambda k \forall q.((q(@) \wedge q(k)) \vee} \\
& \qquad \underline{((q(@) \vee q(k)) \wedge (\lambda \vec{x}.*))) = q(@)\big]}\big) \wedge abt\,(x,p)\big)\big)\big) \twoheadrightarrow p(j)\big) \\
= \; & \lambda i \lambda j.\underline{[\forall q.q(@) \twoheadrightarrow q(j)]} \wedge \big(\forall p.\big(p(i) \wedge \big(\exists x.\big(abt\,\big(x,\underline{q(@)}\big) \wedge abt\,(x,p)\big)\big)\big) \twoheadrightarrow p(j)\big) \\
= \; & \lambda i \lambda j.\underline{[\forall q.q(@) \twoheadrightarrow q(j)]} \wedge \big(\forall p.\big(p(i) \wedge \big(\exists x.(\underline{E(x,@)} \wedge abt\,(x,p))\big)\big) \twoheadrightarrow p(j)\big) \\
= \; & \lambda i \lambda j \underline{\forall p}.\big(\underline{(p(@)} \wedge p(i)) \wedge (\exists x.(\underline{E(x,@)} \wedge abt\,(x,p)))\big)\underline{\twoheadrightarrow p(j)}
\end{aligned}$$

Above, the step from the fourth to the fifth line is justified by our definition of the approximation predicate $\twoheadrightarrow$ from Notation 6.1.1 and by our axioms for the behavior of the aboutness predicate *abt* from Chapter 6.1.

APPENDIX D

PTQ Translations and Definitions

D.1. Solving Partee's Temperature Puzzle

This section shows the possibility of treating Partee's 'temperature puzzle' in an o-based single-type semantics. Partee's puzzle regards the logical – but counterintuitive – derivability of the translation of (45) from the conjunction of the translations of (43) and (44) in an extensional semantics, cf. (**Montague, 1973**, pp. 267–268).

(43) [The temperature$_e$] [is$_{\langle e,\langle e,t\rangle\rangle}$] [ninety$_{\underline{e}}$]

(44) [The temperature$_{\langle s,e\rangle}$] [rises$_{\langle\langle s,e\rangle,t\rangle}$]

(45) [Ninety$_{\underline{\langle s,e\rangle}}$] [rises$_{\langle\langle s,e\rangle,t\rangle}$]

Montague (**1973**) blocks this inference by interpreting intensional common nouns (e.g. temperature) and intransitive verbs (e.g. rise) as functions over *individual concepts* (i.e. as functions over functions from indices to individuals, type $\underline{\langle s,} e\rangle$), and by restricting the interpretation of the verb is to a relation between the *extensions* of two individual concepts at the current index @ (i.e. to a relation between *individuals*). Since (43) thus only asserts the identity of the individual 'the temperature at @' and the number ninety, it blocks the substitution[1] of the 'individual concept'-denoting NP the temperature in (44) by the NP ninety.

However, since our single-type semantics does not command correlates of individual concepts, we cannot directly adopt the above strategy in the 'pure' single-type semantics from Chapter 3. (Similar observations hold for the 'mixed' single-type semantics from Chapters 7 and 8).

Our solution to the temperature puzzle in these semantics is a single-type variant of the puzzle's solution in semantics with the basic types e and $\langle s,t\rangle$.[2] This solution involves an inversion of Montague's strategy of "generalizing to the worst case" (**Partee, 1996**, p. 34): Rather than interpreting all PTQ words as single-

[1] To make the different typing of the two occurrences of the NP ninety in (43) and (45) more perspicuous, we have subscripted the different forms from (43) to (45) by their associated type.

[2] Since the fragment from (**Montague, 1970a**) does not contain intensional nouns or intransitive verbs, it does not give a strategy for their accommodation in an $\{e, \langle s,t\rangle\}$-based semantics.

type correlates of functions over individual *concepts* (codable in the type $\langle\underline{\langle s,t\rangle},e\rangle$; our type $(o;o)$), and obtaining their extensional equivalents through the use of meaning postulates, we interpret them as single-type correlates of functions over *individuals* (our type o). Only intensional common nouns (e.g. temperature, price) and intensional intransitive verbs (rise, change) retain their interpretation in the single-type correlates of functions over individual concepts (here, in the correlate, $((o;o);o)$, of the type $\langle\langle s,e\rangle,t\rangle$[3]). The application of translations of intensional forms to the translations of other PTQ forms is then handled through the use of type- (or meaning-)shifting.

In particular, to enable the application of the TY_0 translations of determiners (type $((o;o)\,(o;o);o)$) to the translations of intensional common nouns (type $(\underline{(o;o)};o)$), we introduce the function *ext* (for '*extensionalization*'). This function sends single-type correlates of properties of individual concepts (coded in the type $\langle\langle\underline{\langle s,t\rangle},e\rangle,\langle s,t\rangle\rangle$) to the correlates of properties of individuals (type $\langle e,\langle s,t\rangle\rangle$). Below, we will use the typing conventions on TY_0 variables from Table 3.4:

Definition D.1.1. The function $ext := \lambda\boldsymbol{Q}\lambda\boldsymbol{x}\bigvee\boldsymbol{P}.\,\boldsymbol{Q}(\boldsymbol{P})\wedge\boldsymbol{x}\doteq\boldsymbol{P}(\boldsymbol{w})$ sends TY_0 terms of the type $(\underline{(o;o)};o)$ to TY_0 terms of the type $(o;o)$.

The function *ext* enables the 'extensionalization' of the TY_0 translation, $\boldsymbol{temp}$, of the noun temperature to the term $\lambda\boldsymbol{x}\bigvee\boldsymbol{P}.\,\boldsymbol{temp}(\boldsymbol{P})\wedge\boldsymbol{x}\doteq\boldsymbol{P}(\boldsymbol{w})$. This term denotes the property of being identical to the result of applying some type-$(o;o)$ witness $\boldsymbol{P}$ of the property denoted by $\boldsymbol{temp}$ to the type-o correlate, $\boldsymbol{w}$, of the current index. As a result, the term $\lambda\boldsymbol{x}\bigvee\boldsymbol{P}.\,\boldsymbol{temp}(\boldsymbol{P})\wedge\boldsymbol{x}\doteq\boldsymbol{P}(\boldsymbol{w})$ denotes the TY_0 correlate of the property of being the temperature at the current index.

The possibility of interpreting intensional common nouns in the type $(o;o)$ enables the TY_0 translation of the logical form of the sentence The temperature is ninety. This translation proceeds as follows:

(3.2.21)

1. $[_{\mathrm{NP}}\mathsf{ninety}] \rightsquigarrow \boldsymbol{ninety}$
2. $[_{\mathrm{TV}}\mathsf{is}] \rightsquigarrow \lambda\boldsymbol{Q}\lambda\boldsymbol{x}.\boldsymbol{Q}(\lambda\boldsymbol{y}.\boldsymbol{x}\doteq\boldsymbol{y})$
3. $[_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{is}][_{\mathrm{NP}}\mathsf{ninety}]] \rightsquigarrow \lambda\boldsymbol{Q}\lambda\boldsymbol{x}.\boldsymbol{Q}(\lambda\boldsymbol{y}.\boldsymbol{x}\doteq\boldsymbol{y})\,[\mathit{lift}(\boldsymbol{ninety})]$
   $= \lambda\boldsymbol{Q}\lambda\boldsymbol{x}.\boldsymbol{Q}(\lambda\boldsymbol{y}.\boldsymbol{x}\doteq\boldsymbol{y})\,[\lambda\boldsymbol{P}.\boldsymbol{P}(\boldsymbol{ninety})]$
   $= \lambda\boldsymbol{x}.[\lambda\boldsymbol{P}.\boldsymbol{P}(\boldsymbol{ninety})](\lambda\boldsymbol{y}.\boldsymbol{x}\doteq\boldsymbol{y})$
   $= \lambda\boldsymbol{x}.[(\lambda\boldsymbol{y}.\boldsymbol{x}\doteq\boldsymbol{y})\,(\boldsymbol{ninety})]$
   $= \lambda\boldsymbol{x}.\boldsymbol{x}\doteq\boldsymbol{ninety}$
4. $[_{\mathrm{N}}\mathsf{temperature}] \rightsquigarrow \boldsymbol{temp} = \lambda\boldsymbol{P}.\boldsymbol{temp}(\boldsymbol{P})$

[3]This type can be coded in the type $\langle\langle\langle s,t\rangle,e\rangle,\langle s,t\rangle\rangle$ in an $\{e,\langle s,t\rangle\}$-based semantics.

5. $[_{\text{DET}}\textsf{the}] \rightsquigarrow \lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}.(\boldsymbol{P}_1(\boldsymbol{y}) \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}) \wedge \boldsymbol{P}(\boldsymbol{x})$

6. $[_{\text{NP}}[_{\text{DET}}\textsf{the}][_{\text{N}}\textsf{temperature}]]$

$$\begin{aligned}
&\rightsquigarrow \lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}.(\boldsymbol{P}_1(\boldsymbol{y}) \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}) \wedge \boldsymbol{P}(\boldsymbol{x}) \, [\mathit{ext}(\boldsymbol{temp})] \\
&= \lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}.(\boldsymbol{P}_1(\boldsymbol{y}) \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}) \wedge \boldsymbol{P}(\boldsymbol{x}) \, \big[\lambda \boldsymbol{z} \bigvee \boldsymbol{P}_2.\, \boldsymbol{temp}(\boldsymbol{P}_2) \wedge \boldsymbol{z} \doteq \boldsymbol{P}_2(\boldsymbol{w})\big] \\
&= \lambda \boldsymbol{P} \bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}.\big(\big[\lambda \boldsymbol{z} \bigvee \boldsymbol{P}_2.\, \boldsymbol{temp}(\boldsymbol{P}_2) \wedge \boldsymbol{z} \doteq \boldsymbol{P}_2(\boldsymbol{w})\big](\boldsymbol{y}) \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}\big) \wedge \boldsymbol{P}(\boldsymbol{x}) \\
&= \lambda \boldsymbol{P} \bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}.\big(\big[\bigvee \boldsymbol{P}_2.\, \boldsymbol{temp}(\boldsymbol{P}_2) \wedge \boldsymbol{y} \doteq \boldsymbol{P}_2(\boldsymbol{w})\big] \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}\big) \wedge \boldsymbol{P}(\boldsymbol{x})
\end{aligned}$$

7. $[_{\text{S}}[_{\text{NP}}[_{\text{DET}}\textsf{the}][_{\text{N}}\textsf{temperature}]][_{\text{VP}}[_{\text{TV}}\textsf{is}][_{\text{NP}}\textsf{ninety}]]]$

$$\begin{aligned}
&\rightsquigarrow \lambda \boldsymbol{P} \bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}.\big(\big[\bigvee \boldsymbol{P}_2.\, \boldsymbol{temp}(\boldsymbol{P}_2) \wedge \boldsymbol{y} \doteq \boldsymbol{P}_2(\boldsymbol{w})\big] \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}\big) \wedge \boldsymbol{P}(\boldsymbol{x}) \\
&\qquad\qquad [\lambda \boldsymbol{z}.\, \boldsymbol{z} \doteq \boldsymbol{ninety}] \\
&= \bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}.\big(\big[\bigvee \boldsymbol{P}.\, \boldsymbol{temp}(\boldsymbol{P}) \wedge \boldsymbol{y} \doteq \boldsymbol{P}(\boldsymbol{w})\big] \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}\big) \wedge [\lambda \boldsymbol{z}.\, \boldsymbol{z} \doteq \boldsymbol{ninety}](\boldsymbol{x}) \\
&= \bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}.\big(\big(\bigvee \boldsymbol{P}.\, \boldsymbol{temp}(\boldsymbol{P}) \wedge \boldsymbol{y} \doteq \boldsymbol{P}(\boldsymbol{w})\big) \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}\big) \wedge \boldsymbol{x} \doteq \boldsymbol{ninety}
\end{aligned}$$

To enable the application of the TY_0 translations of intensional noun phrases (e.g. the temperature; standardly, type $((o;o);o)$) to the translations of intensional intransitive verbs (type $((o;o);o)$), we introduce the function *int* (for '*intensionalization*'). This function sends the single-type correlates of generalized quantifiers over *individuals* (type $((o;o);o)$) to the correlates of generalized quantifiers over individual *concepts* (type $(((\underline{o};o);o);o)$), and sends the single-type correlates of properties of ordered pairs of sets of *individuals* (type $((o;o)\,(o;o);o)$) to the correlates of sets of ordered pairs of sets of individual *concepts* (type $(((\underline{o};o);o)\,((\underline{o};o);o);o)$). Below, we assume that $\boldsymbol{c} \in \{\boldsymbol{john}, \boldsymbol{mary}, \boldsymbol{bill}, \boldsymbol{ninety}, \boldsymbol{sherlock}, \boldsymbol{moriarty}, \boldsymbol{pat}, \boldsymbol{partee}\}$:

Definition D.1.2. The function *int* then operates as follows:

$$\begin{aligned}
\mathit{int}\big(\lambda \boldsymbol{P}.\boldsymbol{P}(\boldsymbol{c})\big) &:= \lambda \boldsymbol{Q} \bigvee \boldsymbol{P}_1.\, \boldsymbol{Q}(\boldsymbol{P}_1) \wedge \boldsymbol{P}_1(\boldsymbol{w}) \doteq \boldsymbol{c} \\
\mathit{int}\big(\lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \bigvee \boldsymbol{x}.\boldsymbol{P}_1(\boldsymbol{x}) \wedge \boldsymbol{P}(\boldsymbol{x})\big) &:= \lambda \boldsymbol{Q}_1 \lambda \boldsymbol{Q} \bigvee \boldsymbol{P}.\, \boldsymbol{Q}_1(\boldsymbol{P}) \wedge \boldsymbol{Q}(\boldsymbol{P}) \\
\mathit{int}\big(\lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \bigwedge \boldsymbol{x}.\boldsymbol{P}_1(\boldsymbol{x}) \dot{\rightarrow} \boldsymbol{P}(\boldsymbol{x})\big) &:= \lambda \boldsymbol{Q}_1 \lambda \boldsymbol{Q} \bigwedge \boldsymbol{P}.\, \boldsymbol{Q}_1(\boldsymbol{P}) \dot{\rightarrow} \boldsymbol{Q}(\boldsymbol{P})
\end{aligned}$$

$$\begin{aligned}
&\mathit{int}\big(\lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}.(\boldsymbol{P}_1(\boldsymbol{y}) \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}) \wedge \boldsymbol{P}(\boldsymbol{x})\big) \\
:= \; &\lambda \boldsymbol{Q}_1 \lambda \boldsymbol{Q} \bigvee \boldsymbol{P} \bigwedge \boldsymbol{P}_1.(\boldsymbol{Q}_1(\boldsymbol{P}_1) \dot{\leftrightarrow} \boldsymbol{P} \doteq \boldsymbol{P}_1) \wedge \boldsymbol{Q}(\boldsymbol{P})
\end{aligned}$$

The function *int* is an 'e-to-$\langle s, e\rangle$'-restricted variant of the intensionalization operation for extensional TY_2 terms from (**van Eijck and Unger, 2010**, Ch. 8.4),

cf. (**Ben-Avi and Winter, 2007**; **de Groote and Kanazawa, 2013**). The latter is a function that systematically replaces each occurrence of the types e and t in the type of a linguistic expression by the types $\langle s, e\rangle$ and $\langle s, t\rangle$, respectively. As a result, the type for generalized quantifiers over individuals, $\langle\langle e, t\rangle, t\rangle$, will be replaced by the type $\langle\langle\langle \underline{s}, e\rangle, \langle s, t\rangle\rangle, \langle s, t\rangle\rangle$. The type for sets of ordered pairs of sets of individuals, $\langle\langle e, t\rangle, \langle\langle e, t\rangle, t\rangle\rangle$, will be replaced by the type $\langle\langle\langle \underline{s}, e\rangle, \langle s, t\rangle\rangle, \langle\langle\langle \underline{s}, e\rangle, \langle s, t\rangle\rangle, \langle s, t\rangle\rangle\rangle$. These types code the types for generalized quantifiers over individual *concepts*, $\langle\langle\langle s, e\rangle, t\rangle, t\rangle$, and for sets of ordered pairs of sets of individual concepts, $\langle\langle\langle s, e\rangle, t\rangle, \langle\langle\langle s, e\rangle, t\rangle, t\rangle\rangle$.

The possibility of interpreting intensional NPs in the type $(((o; o); o); o)$ enables the TY_0 translation of the logical form of the sentence The temperature rises:

(3.2.6)

1. $[_{\mathrm{IV}}\mathsf{rises}] \rightsquigarrow \boldsymbol{rise} = \lambda \boldsymbol{P}.\boldsymbol{rise}(\boldsymbol{P})$
2. $[_{\mathrm{N}}\mathsf{temperature}] \rightsquigarrow \boldsymbol{temp} = \lambda \boldsymbol{P}.\boldsymbol{temp}(\boldsymbol{P})$
3. $[_{\mathrm{DET}}\mathsf{the}] \rightsquigarrow \lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}.(\boldsymbol{P}_1(\boldsymbol{y}) \leftrightarrow \boldsymbol{x} \doteq \boldsymbol{y}) \wedge \boldsymbol{P}(\boldsymbol{x})$
4. $[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{the}][_{\mathrm{N}}\mathsf{temperature}]]$
   $\rightsquigarrow int(\lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}.(\boldsymbol{P}_1(\boldsymbol{y}) \leftrightarrow \boldsymbol{x} \doteq \boldsymbol{y}) \wedge \boldsymbol{P}(\boldsymbol{x}))[\boldsymbol{temp}]$
   $= \lambda \boldsymbol{Q}_1 \lambda \boldsymbol{Q} \bigvee \boldsymbol{P} \bigwedge \boldsymbol{P}_1.(\boldsymbol{Q}_1(\boldsymbol{P}_1) \leftrightarrow \boldsymbol{P} \doteq \boldsymbol{P}_1) \wedge \boldsymbol{Q}(\boldsymbol{P})[\boldsymbol{temp}]$
   $= \lambda \boldsymbol{Q} \bigvee \boldsymbol{P} \bigwedge \boldsymbol{P}_1.(\boldsymbol{temp}(\boldsymbol{P}_1) \leftrightarrow \boldsymbol{P} \doteq \boldsymbol{P}_1) \wedge \boldsymbol{Q}(\boldsymbol{P})$
5. $[_{\mathrm{S}}[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{the}][_{\mathrm{N}}\mathsf{temperature}]][_{\mathrm{IV}}\mathsf{rises}]]$
   $\rightsquigarrow \lambda \boldsymbol{Q} \bigvee \boldsymbol{P} \bigwedge \boldsymbol{P}_1.(\boldsymbol{temp}(\boldsymbol{P}_1) \leftrightarrow \boldsymbol{P} \doteq \boldsymbol{P}_1) \wedge \boldsymbol{Q}(\boldsymbol{P})[\boldsymbol{rise}]$
   $= \bigvee \boldsymbol{P} \bigwedge \boldsymbol{P}_1.(\boldsymbol{temp}(\boldsymbol{P}_1) \leftrightarrow \boldsymbol{P} \doteq \boldsymbol{P}_1) \wedge \boldsymbol{rise}(\boldsymbol{P})$

The TY_0 translation of the sentence A price rises is analogously obtained:

(3.2.5)

1. $[_{\mathrm{IV}}\mathsf{rises}] \rightsquigarrow \boldsymbol{rise} = \lambda \boldsymbol{P}.\boldsymbol{rise}(\boldsymbol{P})$
2. $[_{\mathrm{N}}\mathsf{price}] \rightsquigarrow \boldsymbol{price} = \lambda \boldsymbol{P}.\boldsymbol{price}(\boldsymbol{P})$
3. $[_{\mathrm{DET}}\mathsf{a}] \rightsquigarrow \lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \bigvee \boldsymbol{x}.\boldsymbol{P}_1(\boldsymbol{x}) \wedge \boldsymbol{P}(\boldsymbol{x})$
4. $[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{price}]] \rightsquigarrow int(\lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \bigvee \boldsymbol{x}.\boldsymbol{P}_1(\boldsymbol{x}) \wedge \boldsymbol{P}(\boldsymbol{x}))[\boldsymbol{price}]$
   $= \lambda \boldsymbol{Q}_1 \lambda \boldsymbol{Q} \bigvee \boldsymbol{P}.\boldsymbol{Q}_1(\boldsymbol{P}) \wedge \boldsymbol{Q}(\boldsymbol{P})[\boldsymbol{price}]$
   $= \lambda \boldsymbol{Q} \bigvee \boldsymbol{P}.\boldsymbol{price}(\boldsymbol{P}) \wedge \boldsymbol{Q}(\boldsymbol{P})$
5. $[_{\mathrm{S}}[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{price}]][_{\mathrm{IV}}\mathsf{rises}]] \rightsquigarrow \lambda \boldsymbol{Q} \bigvee \boldsymbol{P}.\boldsymbol{price}(\boldsymbol{P}) \wedge \boldsymbol{Q}(\boldsymbol{P})[\boldsymbol{rise}]$
   $= \bigvee \boldsymbol{P}.\boldsymbol{price}(\boldsymbol{P}) \wedge \boldsymbol{rise}(\boldsymbol{P})$

By using the complex function $\lambda\boldsymbol{x}.int(lift(\boldsymbol{x}))$, we can translate the logical form of the sentence Ninety rises as follows:

(D.1.1)

1. $[_{\text{NP}}\text{ninety}] \rightsquigarrow \boldsymbol{ninety}$
2. $[_{\text{IV}}\text{rises}] \rightsquigarrow \boldsymbol{rise} = \lambda\boldsymbol{P}.\boldsymbol{rise}(\boldsymbol{P})$
3. $[_{\text{S}}[_{\text{NP}}\text{ninety}][_{\text{IV}}\text{rises}]] \rightsquigarrow int(lift(\boldsymbol{ninety}))[\lambda\boldsymbol{P}.\boldsymbol{rise}(\boldsymbol{P})]$
   $= int(\lambda\boldsymbol{P}_1.\boldsymbol{P}_1(\boldsymbol{ninety}))[\lambda\boldsymbol{P}.\boldsymbol{rise}(\boldsymbol{P})]$
   $= \lambda\boldsymbol{Q}\bigvee\boldsymbol{P}.\boldsymbol{Q}(\boldsymbol{P}) \wedge \boldsymbol{P}(\boldsymbol{w}) \doteq \boldsymbol{ninety}[\lambda\boldsymbol{P}_1.\boldsymbol{rise}(\boldsymbol{P}_1)]$
   $= \bigvee\boldsymbol{P}.[\lambda\boldsymbol{P}_1.\boldsymbol{rise}(\boldsymbol{P}_1)](\boldsymbol{P}) \wedge \boldsymbol{P}(\boldsymbol{w}) \doteq \boldsymbol{ninety}$
   $= \bigvee\boldsymbol{P}.\boldsymbol{rise}(\boldsymbol{P}) \wedge \boldsymbol{P}(\boldsymbol{w}) \doteq \boldsymbol{ninety}$

This completes our TY_0 translation of the 'ingredient sentences' for Partee's temperature puzzle. The premise of the intuitively invalid inference from the conjunction of the logical forms from (3.2.21.7) and (3.2.6.5) to the form from (D.1.1.3) is then obtained via the TY_0 translation of the connective and from Definition 3.2.4:

(D.1.2)

1. $[_{\text{S}}[_{\text{NP}}[_{\text{DET}}\text{the}][_{\text{N}}\text{temperature}]][_{\text{VP}}[_{\text{TV}}\text{is}][_{\text{NP}}\text{ninety}]]]$
   $\rightsquigarrow \bigvee\boldsymbol{x}\bigwedge\boldsymbol{y}.\big([\bigvee\boldsymbol{P}_2.\boldsymbol{temp}(\boldsymbol{P}_2) \wedge \boldsymbol{y} \doteq (\boldsymbol{P}_2)(\boldsymbol{w})] \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}\big) \wedge \boldsymbol{x} \doteq \boldsymbol{ninety}$
2. $[_{\text{S}}[_{\text{NP}}[_{\text{DET}}\text{the}][_{\text{N}}\text{temperature}]][_{\text{IV}}\text{rises}]]$
   $\rightsquigarrow \bigvee\boldsymbol{P}\bigwedge\boldsymbol{P}_1.(\boldsymbol{temp}(\boldsymbol{P}_1) \dot{\leftrightarrow} \boldsymbol{P} \doteq \boldsymbol{P}_1) \wedge \boldsymbol{rise}(\boldsymbol{P})$
3. $[_{\text{CONJ}}\text{and}] \rightsquigarrow \lambda\boldsymbol{q}\lambda\boldsymbol{p}.\boldsymbol{p} \wedge \boldsymbol{q}$
4. $[_{\text{S}}[_{\text{S}}[_{\text{NP}}[_{\text{DET}}\text{the}][_{\text{N}}\text{temperature}]][_{\text{VP}}[_{\text{TV}}\text{is}][_{\text{NP}}\text{ninety}]]]$
   $[[_{\text{CONJ}}\text{and}][_{\text{S}}[_{\text{NP}}[_{\text{DET}}\text{the}][_{\text{N}}\text{temperature}]][_{\text{IV}}\text{rises}]]]]$
   $\rightsquigarrow \big(\bigvee\boldsymbol{x}\bigwedge\boldsymbol{y}.\big([\bigvee\boldsymbol{P}_2.\boldsymbol{temp}(\boldsymbol{P}_2) \wedge \boldsymbol{y} \doteq (\boldsymbol{P}_2)(\boldsymbol{w})] \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}\big) \wedge \boldsymbol{x} \doteq \boldsymbol{ninety}\big) \wedge$
   $\big(\bigvee\boldsymbol{P}\bigwedge\boldsymbol{P}_1.(\boldsymbol{temp}(\boldsymbol{P}_1) \dot{\leftrightarrow} \boldsymbol{P} \doteq \boldsymbol{P}_1) \wedge \boldsymbol{rise}(\boldsymbol{P})\big)$

Notably, while the TY_0 term from (D.1.2.2) attributes the property 'rise' to a type-$\langle o;o\rangle$ object that has the property of being a temperature, the TY_0 term from (D.1.2.1) attributes the property 'is ninety' only to the result (type o) of applying a temperature object to the TY_0 correlate of the current index. In virtue of this fact, the logical form from (D.1.1.3) does not follow from the form from (D.1.2.4) by the rules of the logic TY_0.

D.2. 'Weak' PTQ Definitions

This section derives some definitions of the WTY_1^3 translations of the logical sentence forms from Chapter 7.2.2. In particular, the definition of the WTY_1^3 translation of the logical form of the sentence The man walks (cf. (3.2.4)) is obtained as follows:

(7.2.10)

1. $[_{\mathrm{DET}}\mathsf{the}] \rightsquigarrow \lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}.(\boldsymbol{P}_1(\boldsymbol{y}) \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}) \wedge \boldsymbol{P}(\boldsymbol{x})$
   $= \lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \lambda i \exists \boldsymbol{x} \forall \boldsymbol{y}.(\boldsymbol{P}_1(\boldsymbol{y}, i) \leftrightarrow \boldsymbol{x} = \boldsymbol{y}) \wedge \boldsymbol{P}(\boldsymbol{x}, i)$
2. $[_{\mathrm{N}}\mathsf{man}] \rightsquigarrow \boldsymbol{man} = \lambda \boldsymbol{x} \lambda i.\, man\big([\iota x.\boldsymbol{x} = (\lambda j.E(x,j))], i\big)$
3. $[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{the}][_{\mathrm{N}}\mathsf{man}]] \rightsquigarrow \lambda \boldsymbol{P} \bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}.(\boldsymbol{man}(\boldsymbol{y}) \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}) \wedge \boldsymbol{P}(\boldsymbol{x})$
   $= \lambda \boldsymbol{P}_1 \lambda \boldsymbol{P} \lambda i \exists \boldsymbol{x} \forall \boldsymbol{y}.(\boldsymbol{P}_1(\boldsymbol{y}, i) \leftrightarrow \boldsymbol{x} = \boldsymbol{y}) \wedge \boldsymbol{P}(\boldsymbol{x}, i)$
   $\big[\lambda \boldsymbol{z} \lambda k.\, man\big([\iota x.\boldsymbol{z} = (\lambda j.E(x,j))], k\big)\big]$
   $= \lambda \boldsymbol{P} \lambda i \exists \boldsymbol{x} \forall \boldsymbol{y}.\big(\big[\lambda \boldsymbol{z} \lambda k.\, man\big([\iota x.\boldsymbol{z} = (\lambda j.E(x,j))], k\big)\big](\boldsymbol{y}, i) \leftrightarrow \boldsymbol{x} = \boldsymbol{y}\big) \wedge$
   $\boldsymbol{P}(\boldsymbol{x}, i)$
   $= \lambda \boldsymbol{P} \lambda i \exists \boldsymbol{x} \forall \boldsymbol{y}.\big(man\big([\iota x.\boldsymbol{y} = (\lambda j.E(x,j))], i\big) \leftrightarrow \boldsymbol{x} = \boldsymbol{y}\big) \wedge \boldsymbol{P}(\boldsymbol{x}, i)$
4. $[_{\mathrm{VP}}[_{\mathrm{IV}}\mathsf{walks}]] \rightsquigarrow \boldsymbol{walks} = \lambda \boldsymbol{x} \lambda i.\, walk\big([\iota x.\boldsymbol{x} = (\lambda j.E(x,j))], i\big)$
5. $[_{\mathrm{S}}[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{the}][_{\mathrm{N}}\mathsf{man}]][_{\mathrm{VP}}[_{\mathrm{IV}}\mathsf{walks}]]]$
   $\rightsquigarrow \bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}.(\boldsymbol{man}(\boldsymbol{y}) \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}) \wedge \boldsymbol{walk}(\boldsymbol{x})$
   $= \lambda \boldsymbol{P} \lambda i \exists \boldsymbol{x} \forall \boldsymbol{y}.\big(man\big([\iota x.\boldsymbol{y} = (\lambda j.E(x,j))], i\big) \leftrightarrow \boldsymbol{x} = \boldsymbol{y}\big) \wedge \boldsymbol{P}(\boldsymbol{x}, i)$
   $\big[\lambda \boldsymbol{z} \lambda k.\, walk\big([\iota z.\boldsymbol{z} = (\lambda k_1.E(z,k_1))], k\big)\big]$
   $= \lambda i \exists \boldsymbol{x} \forall \boldsymbol{y}.\big(man\big([\iota x.\boldsymbol{y} = (\lambda j.E(x,j))], i\big) \leftrightarrow \boldsymbol{x} = \boldsymbol{y}\big) \wedge$
   $\big[\lambda \boldsymbol{z} \lambda k.\, walk\big([\iota z.\boldsymbol{z} = (\lambda k_1.E(z,k_1))], k\big)\big](\boldsymbol{x}, i)$
   $= \lambda i \exists \boldsymbol{x} \forall \boldsymbol{y}.\big(man\big([\iota x.\boldsymbol{y} = (\lambda j.E(x,j))], i\big) \leftrightarrow \boldsymbol{x} = \boldsymbol{y}\big) \wedge$
   $walk\big([\iota z.\boldsymbol{x} = (\lambda k_1.E(z,k_1))], i\big)$
   $= \lambda i \exists x \forall y.(man(y, i) \leftrightarrow x = y) \wedge walk(x, i)$

The definition of the WTY_1^3 translation of the logical form of the sentence A price rises (cf. (3.2.5)) is given below:

(7.2.11)

1. $[_{\mathrm{IV}}\mathsf{rises}] \rightsquigarrow \boldsymbol{rise} = \lambda\boldsymbol{P}.\boldsymbol{rise}(\boldsymbol{P})$
   $= \lambda\boldsymbol{P}\lambda i.\mathit{rise}\big(\big[\iota T.\big(\forall \boldsymbol{z}.\boldsymbol{P}(\boldsymbol{z}) = [\lambda j.E(T(\boldsymbol{z}), j)]\big)\big], i\big)$
2. $[_{\mathrm{N}}\mathsf{price}] \rightsquigarrow \boldsymbol{price} = \lambda\boldsymbol{P}.\boldsymbol{price}(\boldsymbol{P})$
   $= \lambda\boldsymbol{P}\lambda i.\mathit{price}\big(\big[\iota T.\big(\forall \boldsymbol{z}.\boldsymbol{P}(\boldsymbol{z}) = [\lambda j.E(T(\boldsymbol{z}), j)]\big)\big], i\big)$
3. $[_{\mathrm{DET}}\mathsf{a}] \rightsquigarrow \lambda\boldsymbol{P}_1\lambda\boldsymbol{P}\bigvee \boldsymbol{x}.\boldsymbol{P}_1(\boldsymbol{x}) \wedge \boldsymbol{P}(\boldsymbol{x}) = \lambda\boldsymbol{P}_1\lambda\boldsymbol{P}\lambda i\exists \boldsymbol{x}.\boldsymbol{P}_1(\boldsymbol{x}, i) \wedge \boldsymbol{P}(\boldsymbol{x}, i)$
4. $[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{price}]] \rightsquigarrow \lambda\boldsymbol{Q}\bigvee \boldsymbol{P}.\boldsymbol{price}(\boldsymbol{P}) \wedge \boldsymbol{Q}(\boldsymbol{P})$
   $= \lambda\boldsymbol{Q}_1\lambda\boldsymbol{Q}\lambda i\exists \boldsymbol{P}.\boldsymbol{Q}_1(\boldsymbol{P}, i) \wedge \boldsymbol{Q}(\boldsymbol{P}, i)\big[\lambda\boldsymbol{P}_1\lambda j.\mathit{price}\big(\big[\iota T.\big(\forall \boldsymbol{z}.\boldsymbol{P}_1(\boldsymbol{z}) =$
   $[\lambda k.E(T(\boldsymbol{z}), k)]\big)\big], j\big)\big]$
   $= \lambda\boldsymbol{Q}\lambda i\exists \boldsymbol{P}.\big[\lambda\boldsymbol{P}_1\lambda j.\mathit{price}\big(\big[\iota T.\big(\forall \boldsymbol{z}.\boldsymbol{P}_1(\boldsymbol{z}) = [\lambda k.E(T(\boldsymbol{z}), k)]\big)\big], j\big)\big](\boldsymbol{P}, i) \wedge \boldsymbol{Q}(\boldsymbol{P}, i)$
   $= \lambda\boldsymbol{Q}\lambda i\exists \boldsymbol{P}.\mathit{price}\big(\big[\iota T.\big(\forall \boldsymbol{z}.\boldsymbol{P}(\boldsymbol{z}) = [\lambda j.E(T(\boldsymbol{z}), j)]\big)\big], i\big) \wedge \boldsymbol{Q}(\boldsymbol{P}, i)$
5. $[_{\mathrm{S}}[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{price}]][_{\mathrm{IV}}\mathsf{rises}]] \rightsquigarrow \bigvee \boldsymbol{P}.\boldsymbol{price}(\boldsymbol{P}) \wedge \boldsymbol{rise}(\boldsymbol{P})$
   $= \lambda\boldsymbol{Q}\lambda i\exists \boldsymbol{P}.\mathit{price}\big(\big[\iota T.\big(\forall \boldsymbol{z}.\boldsymbol{P}(\boldsymbol{z}) = [\lambda j.E(T(\boldsymbol{z}), j)]\big)\big], i\big) \wedge \boldsymbol{Q}(\boldsymbol{P}, i)$
   $\big[\lambda\boldsymbol{P}_1\lambda k.\mathit{rise}\big(\big[\iota T_1.\big(\forall \boldsymbol{y}.\boldsymbol{P}_1(\boldsymbol{y}) = [\lambda k_1.E(T_1(\boldsymbol{y}), k_1)]\big)\big], k\big)\big]$
   $= \lambda i\exists \boldsymbol{P}.\mathit{price}\big(\big[\iota T.\big(\forall \boldsymbol{z}.\boldsymbol{P}(\boldsymbol{z}) = [\lambda j.E(T(\boldsymbol{z}), j)]\big)\big], i\big) \wedge$
   $\big[\lambda\boldsymbol{P}_1\lambda k.\mathit{rise}\big(\big[\iota T_1.\big(\forall \boldsymbol{y}.\boldsymbol{P}_1(\boldsymbol{y}) = [\lambda k_1.E(T_1(\boldsymbol{y}), k_1)]\big)\big], k\big)\big](\boldsymbol{P}, i)$
   $= \lambda i\exists \boldsymbol{P}.\mathit{price}\big(\big[\iota T.\big(\forall \boldsymbol{z}.\boldsymbol{P}(\boldsymbol{z}) = [\lambda j.E(T(\boldsymbol{z}), j)]\big)\big], i\big) \wedge$
   $\mathit{rise}\big(\big[\iota T_1.\big(\forall \boldsymbol{y}.\boldsymbol{P}(\boldsymbol{y}) = [\lambda k.E(T_1(\boldsymbol{y}), k)]\big)\big], i\big)$
   $= \lambda i\exists T.\mathit{price}(T, i) \wedge \mathit{rise}(T, i)$

The WTY_1^3 translation of the logical form of the sentence John finds a unicorn (cf. (3.2.7)) has the following definition:

(7.2.8)

1. $[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{unicorn}]] \rightsquigarrow \lambda\boldsymbol{P}\bigvee \boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{P}(\boldsymbol{x})$
   $= \lambda\boldsymbol{P}\lambda i\exists\boldsymbol{x}.unicorn\big([\iota x.\boldsymbol{x} = (\lambda j.E(x,j))],i\big) \wedge \boldsymbol{P}(\boldsymbol{x},i)$
2. $[_{\mathrm{TV}}\mathsf{finds}] \rightsquigarrow \lambda\boldsymbol{Q}\lambda\boldsymbol{x}.\boldsymbol{Q}(\lambda\boldsymbol{y}.\boldsymbol{find}(\boldsymbol{y},\boldsymbol{x}))$
   $= \lambda\boldsymbol{Q}\lambda\boldsymbol{x}.\boldsymbol{Q}\big(\lambda\boldsymbol{y}\lambda i.find\big([\iota y.\boldsymbol{y} = (\lambda j.E(y,j))],[\iota x.\boldsymbol{x} = (\lambda k.E(x,k))],i\big)\big)$
3. $[_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{finds}][_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{unicorn}]] \rightsquigarrow \lambda\boldsymbol{y}\bigvee \boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{find}(\boldsymbol{x},\boldsymbol{y})$
   $= \lambda\boldsymbol{Q}\lambda\boldsymbol{y}.\boldsymbol{Q}\big(\lambda\boldsymbol{z}\lambda k.find\big([\iota z.\boldsymbol{z} = (\lambda k_1.E(z,k_1))],[\iota y.\boldsymbol{y} = (\lambda k_2.E(y,k_2))],k\big)\big)$
   $\big[\lambda\boldsymbol{P}\lambda i\exists\boldsymbol{x}.unicorn\big([\iota x.\boldsymbol{x} = (j.E(x,j))],i\big) \wedge \boldsymbol{P}(\boldsymbol{x},i)\big]$
   $= \lambda\boldsymbol{y}.\big[\lambda\boldsymbol{P}\lambda i\exists\boldsymbol{x}.unicorn\big([\iota x.\boldsymbol{x} = (j.E(x,j))],i\big) \wedge \boldsymbol{P}(\boldsymbol{x},i)\big]$
   $\big(\lambda\boldsymbol{z}\lambda k.find\big([\iota z.\boldsymbol{z} = (\lambda k_1.E(z,k_1))],[\iota y.\boldsymbol{y} = (\lambda k_2.E(y,k_2))],k\big)\big)$
   $= \lambda\boldsymbol{y}.\big[\lambda i\exists\boldsymbol{x}.unicorn\big([\iota x.\boldsymbol{x} = (j.E(x,j))],i\big) \wedge$
   $\big(\lambda\boldsymbol{z}\lambda k.find\big([\iota z.\boldsymbol{z} = (\lambda k_1.E(z,k_1))],[\iota z.\boldsymbol{y} = (\lambda k_2.E(y,k_2))],k\big)\big)(\boldsymbol{x},i)\big]$
   $= \lambda\boldsymbol{y}\lambda i\exists\boldsymbol{x}.unicorn\big([\iota x.\boldsymbol{x} = (j.E(x,j))],i\big) \wedge$
   $find\big([\iota x.\boldsymbol{x} = (\lambda k.E(x,k))],[\iota y.\boldsymbol{y} = (\lambda k.E(y,k))],i\big)$
   $= \lambda\boldsymbol{y}\lambda i\exists x.unicorn(x,i) \wedge find\big(x,[\iota y.\boldsymbol{y} = (\lambda k.E(y,k))],i\big)$
4. $[_{\mathrm{NP}}\mathsf{John}] \rightsquigarrow \boldsymbol{john} = \lambda i.E(john,i)$
5. $[_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{John}][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{finds}][_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{unicorn}]]]] \rightsquigarrow \bigvee \boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{find}(\boldsymbol{x},\boldsymbol{john})$
   $= \lambda\boldsymbol{y}\lambda i\exists x.unicorn(x,i) \wedge find\big(x,\big[\iota y.\boldsymbol{y} = (\lambda k.E(y,k))\big],i\big)\,[\lambda j.E(john,j)]$
   $= \lambda i\exists x.unicorn(x,i) \wedge find\big(x,\big[\iota y.[\lambda j.E(john,j)] = (\lambda k.E(y,k))\big],i\big)$
   $= \lambda i\exists x.unicorn(x,i) \wedge find(x,john,i)$

The definition of the WTY_1^3 translation of the narrow-scope reading of the sentence John seeks a unicorn (cf. (3.2.13)) is given below. In the definition, X abbreviates the TY_2^3 term from Definition 7.2.2:

(7.2.17)

1. $[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{unicorn}]] \rightsquigarrow \lambda\boldsymbol{P}\bigvee \boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{P}(\boldsymbol{x})$
   $= \lambda\boldsymbol{P}\lambda i\exists\boldsymbol{x}.unicorn\big([\iota x.\boldsymbol{x} = (\lambda j.E(x,j))],i\big) \wedge \boldsymbol{P}(\boldsymbol{x},i)$
2. $[_{\mathrm{TV}}\mathsf{seeks}] \rightsquigarrow \boldsymbol{seek} = \lambda\boldsymbol{Q}\lambda\boldsymbol{x}.\boldsymbol{seek}(\boldsymbol{Q},\boldsymbol{x})$
   $= \lambda\boldsymbol{Q}\,\lambda\boldsymbol{x}\,\lambda i.seek\big([\iota Q.(\forall\boldsymbol{P}.(\lambda k.\,\boldsymbol{Q}(\boldsymbol{P},k)) = (\lambda k_3.Q(X,k_3)))],$
   $[\iota x.\boldsymbol{x} = (\lambda j.E(x,j))],i\big)$
3. $[_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{seeks}][_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{unicorn}]]]$
   $\rightsquigarrow \lambda\boldsymbol{x}.\boldsymbol{seek}\big([\lambda\boldsymbol{P}\bigvee \boldsymbol{y}.\boldsymbol{unicorn}(\boldsymbol{y}) \wedge \boldsymbol{P}(\boldsymbol{y})],\boldsymbol{x}\big)$
   $= \lambda\boldsymbol{Q}\,\lambda\boldsymbol{y}\,\lambda i.seek\big([\iota Q.(\forall\boldsymbol{P}.(\lambda k.\,\boldsymbol{Q}(\boldsymbol{P},k)) = (\lambda k_3.Q(X,k_3)))],$
   $[\iota y.\boldsymbol{y} = (\lambda j.E(y,j))],i\big)$
   $\big[\lambda\boldsymbol{P}_1\lambda k_4\exists\boldsymbol{x}.unicorn\big([\iota x.\boldsymbol{x} = (\lambda k_5.E(x,k_5))],k_4\big) \wedge \boldsymbol{P}_1(\boldsymbol{x},k_4)\big]$
   $= \lambda\boldsymbol{y}\,\lambda i.seek\big(\big[\iota Q.\big(\forall\boldsymbol{P}.\big(\lambda k.\big[\lambda\boldsymbol{P}_1\lambda k_4\exists\boldsymbol{x}.unic.\big([\iota x.\boldsymbol{x} = (\lambda k_5.E(x,k_5))],k_4\big) \wedge$
   $\boldsymbol{P}_1(\boldsymbol{x},k_4)\big](\boldsymbol{P},k)\big) = (\lambda k_3.Q(X,k_3))\big)\big],[\iota y.\boldsymbol{y} = (\lambda j.E(y,j))],i\big)$
   $= \lambda\boldsymbol{y}\,\lambda i.seek\big(\big[\iota Q.\big(\forall\boldsymbol{P}.\big(\lambda k\exists\boldsymbol{x}.unicorn\big([\iota x.\boldsymbol{x} = (\lambda k_5.E(x,k_5))],k\big) \wedge$
   $\boldsymbol{P}(\boldsymbol{x},k)\big) = (\lambda k_3.Q(X,k_3))\big)\big],[\iota y.\boldsymbol{y} = (\lambda j.E(y,j))],i\big)$
   $= \lambda\boldsymbol{y}\,\lambda i.seek\big([\lambda P\lambda k\exists x.unicorn(x,k) \wedge P(x,k)],[\iota y.\boldsymbol{y} = (\lambda j.E(y,j))],i\big)$
4. $[_{\mathrm{NP}}\mathsf{John}] \rightsquigarrow \boldsymbol{john} = \lambda i.E(john,i)$
5. $[_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{John}][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{seeks}][_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{unicorn}]]]]$
   $\rightsquigarrow \boldsymbol{seek}\big([\lambda\boldsymbol{P}\bigvee \boldsymbol{y}.\boldsymbol{unicorn}(\boldsymbol{y}) \wedge \boldsymbol{P}(\boldsymbol{y})],\boldsymbol{john}\big)$
   $= \lambda\boldsymbol{y}\,\lambda i.seek\big([\lambda P\lambda k\exists x.unicorn(x,k) \wedge P(x,k)],$
   $[\iota y.\boldsymbol{y} = (\lambda j.E(y,j))],i\big)[\lambda k_1.E(john,k_1)]$
   $= \lambda i.seek\big([\lambda P\lambda k\exists x.unicorn(x,k) \wedge P(x,k)],$
   $\big[\iota y.[\lambda k_1.E(john,k_1)] = (\lambda j.E(y,j))\big],i\big)$
   $= \lambda i.seek\big([\lambda P\lambda j\exists x.unicorn(x,j) \wedge P(x,j)],john,i\big)$

The WTY_1^3 translation of the sentence's wide-scope reading (cf. (3.2.14)) has the following definition:

(7.2.18)

1. $[_{\text{TV}}\textsf{seeks}] \rightsquigarrow \boldsymbol{seek} = \lambda\boldsymbol{Q}\lambda\boldsymbol{x}.\boldsymbol{seek}(\boldsymbol{Q},\boldsymbol{x})$
   $= \lambda\boldsymbol{Q}\lambda\boldsymbol{x}\lambda i.seek\big([\iota Q.(\forall\boldsymbol{P}.(\lambda k.\boldsymbol{Q}(\boldsymbol{P},k)) = (\lambda k_3.Q(X,k_3)))],$
   $[\iota x.\boldsymbol{x} = (\lambda j.E(x,j))],i\big)$
2. $t_0 \rightsquigarrow lift(\boldsymbol{x}_0) = \lambda\boldsymbol{P}.\boldsymbol{P}(\boldsymbol{x}_0) = \lambda\boldsymbol{P}\lambda i.\boldsymbol{P}(\boldsymbol{x}_0,i)$
3. $[_{\text{VP}}[_{\text{TV}}\textsf{seeks}]\ t_0] \rightsquigarrow \lambda\boldsymbol{x}.\boldsymbol{seek}\big([\lambda\boldsymbol{P}.\boldsymbol{P}(\boldsymbol{x}_0)],\boldsymbol{x}\big)$
   $= \lambda\boldsymbol{Q}\lambda\boldsymbol{x}\lambda i.seek\big([\iota Q.(\forall\boldsymbol{P}.(\lambda k.\boldsymbol{Q}(\boldsymbol{P},k)) = (\lambda k_3.Q(X,k_3)))],$
   $[\iota x.\boldsymbol{x} = (\lambda j.E(x,j))],i\big)\ \big[\lambda\boldsymbol{P}_1\lambda k_4.\boldsymbol{P}_1(\boldsymbol{x}_0,k_4)\big]$
   $= \lambda\boldsymbol{x}\lambda i.seek\big(\big[\iota Q.\big(\forall\boldsymbol{P}.\big(\lambda k.\big[\lambda\boldsymbol{P}_1\lambda k_4.\boldsymbol{P}_1(\boldsymbol{x}_0,k_4)\big](\boldsymbol{P},k)\big) = (\lambda k_3.Q(X,k_3))\big)\big],$
   $[\iota x.\boldsymbol{x} = (\lambda j.E(x,j))],i\big)$
   $= \lambda\boldsymbol{x}\lambda i.seek\big(\big[\iota Q.\big(\forall\boldsymbol{P}.\big(\lambda k.\boldsymbol{P}(\boldsymbol{x}_0,k)\big) = (\lambda k_3.Q(X,k_3))\big)\big],$
   $\big[\iota x.\boldsymbol{x} = (\lambda j.E(x,j))\big],i\big)$
4. $[_{\text{S}}[_{\text{NP}}\textsf{John}][_{\text{VP}}[_{\text{TV}}\textsf{seeks}]\ t_0]] \rightsquigarrow \boldsymbol{seek}\big([\lambda\boldsymbol{P}.\boldsymbol{P}(\boldsymbol{x}_0)],\boldsymbol{john}\big)$
   $= \lambda\boldsymbol{x}\lambda i.seek\big(\big[\iota Q.\big(\forall\boldsymbol{P}.\big(\lambda k.\boldsymbol{P}(\boldsymbol{x}_0,k)\big) = (\lambda k_3.Q(X,k_3))\big)\big],$
   $\big[\iota x.\boldsymbol{x} = (\lambda j.E(x,j))\big],i\big)\big[\lambda k_4.E(john,k_4)\big]$
   $= \lambda i.seek\big(\big[\iota Q.\big(\forall\boldsymbol{P}.\big(\lambda k.\boldsymbol{P}(\boldsymbol{x}_0,k)\big) = (\lambda k_3.Q(X,k_3))\big)\big],$
   $\big[\iota x.\big[\lambda k_4.E(john,k_4)\big] = (\lambda j.E(x,j))\big],i\big)$
   $= \lambda i.seek\big(\big[\iota Q.\big(\forall\boldsymbol{P}.\big(\lambda k.\boldsymbol{P}(\boldsymbol{x}_0,k)\big) = (\lambda k_3.Q(X,k_3))\big)\big],john,i\big)$
5. $[_{\text{NP}}[_{\text{DET}}\textsf{a}][_{\text{N}}\textsf{unicorn}]]^0 \rightsquigarrow \lambda\boldsymbol{P}\bigvee\boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{P}(\boldsymbol{x})$
   $= \lambda\boldsymbol{P}\lambda i\exists\boldsymbol{x}.unicorn\big([\iota x.\boldsymbol{x} = (\lambda j.E(x,j))],i\big) \wedge \boldsymbol{P}(\boldsymbol{x},i)$
6. $[_{\text{S}}[_{\text{NP}}[_{\text{DET}}\textsf{a}][_{\text{N}}\textsf{unicorn}]]^0\ [_{\text{S}}[_{\text{NP}}\textsf{John}][_{\text{VP}}[_{\text{TV}}\textsf{seeks}]\ t_0]]]$
   $\rightsquigarrow \bigvee\boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{seek}\big([\lambda\boldsymbol{P}.\boldsymbol{P}(\boldsymbol{x})],\boldsymbol{john}\big)$
   $= \lambda\boldsymbol{P}\lambda i\exists\boldsymbol{x}.unicorn\big([\iota x.\boldsymbol{x} = (\lambda j.E(x,j))],i\big) \wedge \boldsymbol{P}(\boldsymbol{x},i)$
   $\big[\lambda\boldsymbol{x}_0\lambda k_4.seek\big(\big[\iota Q.\big(\forall\boldsymbol{P}.\big(\lambda k.\boldsymbol{P}(\boldsymbol{x}_0,k)\big) = (\lambda k_3.Q(X,k_3))\big)\big],john,k_4\big)\big]$
   $= \lambda i\exists\boldsymbol{x}.unicorn\big([\iota x.\boldsymbol{x} = (\lambda j.E(x,j))],i\big) \wedge \big[\lambda\boldsymbol{x}_0\lambda k_4.$
   $seek\big(\big[\iota Q.\big(\forall\boldsymbol{P}.\big(\lambda k.\boldsymbol{P}(\boldsymbol{x}_0,k)\big) = (\lambda k_3.Q(X,k_3))\big)\big],john,k_4\big)\big](\boldsymbol{x},i)$
   $= \lambda i\exists\boldsymbol{x}.unicorn\big([\iota x.\boldsymbol{x} = (\lambda j.E(x,j))],i\big) \wedge$
   $seek\big(\big[\iota Q.\big(\forall\boldsymbol{P}.\big(\lambda k.\boldsymbol{P}(\boldsymbol{x},k)\big) = (\lambda k_3.Q(X,k_3))\big)\big],john,i\big)$
   $= \lambda i\exists x.unicorn(x,i) \wedge seek\big([\lambda P\lambda j.P(x,j)],john,i\big)$

The definition of the WTY_1^3 translation of the logical form of the sentence The temperature is ninety (cf. (3.2.21)) runs as follows:

(7.2.25)

1. $[_{\mathrm{NP}}\mathsf{ninety}] \rightsquigarrow \boldsymbol{ninety} = \lambda i.E(ninety, i)$
2. $[_{\mathrm{TV}}\mathsf{is}] \rightsquigarrow \lambda\boldsymbol{Q}\lambda\boldsymbol{x}.\boldsymbol{Q}(\lambda\boldsymbol{y}.\boldsymbol{x} \doteq \boldsymbol{y}) = \lambda\boldsymbol{Q}\lambda\boldsymbol{x}.\boldsymbol{Q}(\lambda\boldsymbol{y}\lambda i.\boldsymbol{x}(i) = \boldsymbol{y}(i))$
3. $[_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{is}][_{\mathrm{NP}}\mathsf{ninety}]] \rightsquigarrow \lambda\boldsymbol{Q}\lambda\boldsymbol{x}.\boldsymbol{Q}(\lambda\boldsymbol{y}.\boldsymbol{x} \doteq \boldsymbol{y})\,[\mathit{lift}(\boldsymbol{ninety})]$
   $= \lambda\boldsymbol{Q}\lambda\boldsymbol{x}.\boldsymbol{Q}(\lambda\boldsymbol{y}\lambda i.\boldsymbol{x}(i) = \boldsymbol{y}(i))\,[\lambda\boldsymbol{P}.\boldsymbol{P}(\lambda j.E(ninety, j))]$
   $= \lambda\boldsymbol{x}.[\lambda\boldsymbol{P}.\boldsymbol{P}(\lambda j.E(ninety, j))](\lambda\boldsymbol{y}\lambda i.\boldsymbol{x}(i) = \boldsymbol{y}(i))$
   $= \lambda\boldsymbol{x}.(\lambda\boldsymbol{y}\lambda i.\boldsymbol{x}(i) = \boldsymbol{y}(i))(\lambda j.E(ninety, j))$
   $= \lambda\boldsymbol{x}\lambda i.\boldsymbol{x}(i) = (\lambda j.E(ninety, j))(i)$
   $= \lambda\boldsymbol{x}\lambda i.\boldsymbol{x}(i) = E(ninety, i)$
4. $[_{\mathrm{N}}\mathsf{temperature}] \rightsquigarrow \boldsymbol{temp} = \lambda\boldsymbol{P}.\boldsymbol{temp}(\boldsymbol{P}) = \lambda\boldsymbol{P}\lambda i.temp\left(\left[\iota T.\left(\forall\boldsymbol{z}.\boldsymbol{P}(\boldsymbol{z}) = [\lambda j.E(T(\boldsymbol{z}), j)]\right)\right], i\right)$
5. $[_{\mathrm{DET}}\mathsf{the}] \rightsquigarrow \lambda\boldsymbol{P}_1\lambda\boldsymbol{P}\bigvee\boldsymbol{x}\bigwedge\boldsymbol{y}.(\boldsymbol{P}_1(\boldsymbol{y}) \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}) \wedge \boldsymbol{P}(\boldsymbol{x})$
   $= \lambda\boldsymbol{P}_1\lambda\boldsymbol{P}\lambda i\exists\boldsymbol{x}\forall\boldsymbol{y}.(\boldsymbol{P}_1(\boldsymbol{y}, i) \leftrightarrow \boldsymbol{x}(i) = \boldsymbol{y}(i)) \wedge \boldsymbol{P}(\boldsymbol{x}, i)$
6. $[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{the}][_{\mathrm{N}}\mathsf{temperature}]] \rightsquigarrow \lambda\boldsymbol{P}\bigvee\boldsymbol{x}\bigwedge\boldsymbol{y}.\left(\boldsymbol{y} \doteq \left(\imath\boldsymbol{P}_2.\boldsymbol{temp}(\boldsymbol{P}_2)\right)(\boldsymbol{w}) \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}\right) \barwedge \boldsymbol{P}(\boldsymbol{x})$
   $= \lambda\boldsymbol{P}_1\lambda\boldsymbol{P}\lambda i\exists\boldsymbol{x}\forall\boldsymbol{y}.(\boldsymbol{P}_1(\boldsymbol{y}, i) \leftrightarrow \boldsymbol{x}(i) = \boldsymbol{y}(i)) \wedge \boldsymbol{P}(\boldsymbol{x}, i)\left[\lambda\boldsymbol{z}\lambda j\exists\boldsymbol{P}_2.\boldsymbol{temp}(\boldsymbol{P}_2, j) \wedge (\lambda k.\boldsymbol{z}(k) = (\boldsymbol{P}_2)(\boldsymbol{w}, k))(j)\right]$
   $= \lambda\boldsymbol{P}_1\lambda\boldsymbol{P}\lambda i\exists\boldsymbol{x}\forall\boldsymbol{y}.(\boldsymbol{P}_1(\boldsymbol{y}, i) \leftrightarrow \boldsymbol{x}(i) = \boldsymbol{y}(i)) \wedge \boldsymbol{P}(\boldsymbol{x}, i)\left[\lambda\boldsymbol{z}\lambda j\exists\boldsymbol{P}_2.\boldsymbol{temp}(\boldsymbol{P}_2, j) \wedge \boldsymbol{z}(j) = (\boldsymbol{P}_2)(\boldsymbol{w}, j)\right]$
   $= \lambda\boldsymbol{P}_1\lambda\boldsymbol{P}\lambda i\exists\boldsymbol{x}\forall\boldsymbol{y}.(\boldsymbol{P}_1(\boldsymbol{y}, i) \leftrightarrow \boldsymbol{x}(i) = \boldsymbol{y}(i)) \wedge \boldsymbol{P}(\boldsymbol{x}, i)\left[\lambda\boldsymbol{z}\lambda j\exists\boldsymbol{P}_2.\right.$
   $\left.\left[\lambda\boldsymbol{P}_3\lambda k.temp\left(\left[\iota T.\left(\forall\boldsymbol{x}_1.\boldsymbol{P}_3(\boldsymbol{x}_1) = [\lambda k_1.E(T(\boldsymbol{x}_1), k_1)]\right)\right], k\right)\right](\boldsymbol{P}_2, j) \wedge \boldsymbol{z}(j) = (\boldsymbol{P}_2)(\boldsymbol{w}, j)\right]$
   $= \lambda\boldsymbol{P}_1\lambda\boldsymbol{P}\lambda i\exists\boldsymbol{x}\forall\boldsymbol{y}.(\boldsymbol{P}_1(\boldsymbol{y}, i) \leftrightarrow \boldsymbol{x}(i) = \boldsymbol{y}(i)) \wedge \boldsymbol{P}(\boldsymbol{x}, i)$
   $\left[\lambda\boldsymbol{z}\lambda j\exists\boldsymbol{P}_2.temp\left(\left[\iota T.\left(\forall\boldsymbol{x}_1.\boldsymbol{P}_2(\boldsymbol{x}_1) = [\lambda k_1.E(T(\boldsymbol{x}_1), k_1)]\right)\right], j\right) \wedge \boldsymbol{z}(j) = (\boldsymbol{P}_2)(\boldsymbol{w}, j)\right]$

$$= \lambda \boldsymbol{P} \lambda i \exists \boldsymbol{x} \forall \boldsymbol{y}. \big(\big[\lambda \boldsymbol{z} \lambda j \exists \boldsymbol{P}_2. temp \big(\big[\iota T. \big(\forall \boldsymbol{x}_1. \boldsymbol{P}_2(\boldsymbol{x}_1) = [\lambda k_1. E(T(\boldsymbol{x}_1), k_1)] \big) \big], j \big) \wedge \boldsymbol{z}(i) = (\boldsymbol{P}_2)(\boldsymbol{w}, i) \big] (\boldsymbol{y}, i) \leftrightarrow \boldsymbol{x}(i) = \boldsymbol{y}(i) \big) \wedge \boldsymbol{P}(\boldsymbol{x}, i)$$

$$= \lambda \boldsymbol{P} \lambda i \exists \boldsymbol{x} \forall \boldsymbol{y}. \big(\big(\exists \boldsymbol{P}_2. temp \big(\big[\iota T. \big(\forall \boldsymbol{x}_1. \boldsymbol{P}_2(\boldsymbol{x}_1) = [\lambda k_1. E(T(\boldsymbol{x}_1), k_1)] \big) \big], i \big) \wedge \boldsymbol{y}(i) = (\boldsymbol{P}_2)(\boldsymbol{w}, i) \big) \leftrightarrow \boldsymbol{x}(i) = \boldsymbol{y}(i) \big) \wedge \boldsymbol{P}(\boldsymbol{x}, i)$$

7. $[_{\text{S}}[_{\text{NP}}[_{\text{DET}}\textsf{the}][_{\text{N}}\textsf{temperature}]][_{\text{VP}}[_{\text{TV}}\textsf{is}][_{\text{NP}}\textsf{ninety}]]]$

$$\rightsquigarrow \bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}. \big(\big(\bigvee \boldsymbol{P}. \boldsymbol{temp}(\boldsymbol{P}) \wedge \boldsymbol{y} \doteq \boldsymbol{P}(\boldsymbol{w}) \big) \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y} \big) \wedge \boldsymbol{x} \doteq \boldsymbol{ninety}$$

$$= \lambda \boldsymbol{P} \lambda i \exists \boldsymbol{x} \forall \boldsymbol{y}. \big(\big(\exists \boldsymbol{P}_2. temp \big(\big[\iota T. \big(\forall \boldsymbol{x}_1. \boldsymbol{P}_2(\boldsymbol{x}_1) = [\lambda k. E(T(\boldsymbol{x}_1), k)] \big) \big], i \big) \wedge \boldsymbol{y}(i) = (\boldsymbol{P}_2)(\boldsymbol{w}, i) \big) \leftrightarrow \boldsymbol{x}(i) = \boldsymbol{y}(i) \big) \wedge \boldsymbol{P}(\boldsymbol{x}, i)[\lambda \boldsymbol{z} \lambda k_1. \boldsymbol{z}(k_1) = E(ninety, k_1)]$$

$$= \lambda i \exists \boldsymbol{x} \forall \boldsymbol{y}. \big(\big(\exists \boldsymbol{P}_2. temp \big(\big[\iota T. \big(\forall \boldsymbol{x}_1. \boldsymbol{P}_2(\boldsymbol{x}_1) = [\lambda k. E(T(\boldsymbol{x}_1), k)] \big) \big], i \big) \wedge \boldsymbol{y}(i) = (\boldsymbol{P}_2)(\boldsymbol{w}, i) \big) \leftrightarrow \boldsymbol{x}(i) = \boldsymbol{y}(i) \big) \wedge [\lambda \boldsymbol{z} \lambda k_1. \boldsymbol{z}(k_1) = E(ninety, k_1)] (\boldsymbol{x}, i)$$

$$= \lambda i \exists \boldsymbol{x} \forall \boldsymbol{y}. \big(\big(\exists \boldsymbol{P}_2. temp \big(\big[\iota T. \big(\forall \boldsymbol{x}_1. \boldsymbol{P}_2(\boldsymbol{x}_1) = [\lambda k. E(T(\boldsymbol{x}_1), k)] \big) \big], i \big) \wedge \boldsymbol{y}(i) = (\boldsymbol{P}_2)(\boldsymbol{w}, i) \big) \leftrightarrow \boldsymbol{x}(i) = \boldsymbol{y}(i) \big) \wedge \boldsymbol{x}(i) = E(ninety, i)$$

$$= \lambda i \exists x \forall y. \big(\big(\exists T. temp(T, i) \wedge y = T(\boldsymbol{w}) \big) \leftrightarrow x = y \big) \wedge x = ninety$$

$$= \lambda i \exists x \forall y. \big(\big(\exists T. temp(T, i) \wedge y = T\big([\lambda j \forall p. p(@) \twoheadrightarrow p(j)] \big) \big) \leftrightarrow x = y \big) \wedge x = ninety$$

The WTY_1^3 translation of the logical form of the sentence John finds and eats a unicorn (cf. (3.2.24)) has the following definition:

(7.2.31)

1. $[_{\mathrm{TV}}\mathsf{finds}] \rightsquigarrow \lambda \boldsymbol{Q} \lambda \boldsymbol{x}. \boldsymbol{Q}(\lambda \boldsymbol{y}. \boldsymbol{find}(\boldsymbol{y}, \boldsymbol{x}))$
   $= \lambda \boldsymbol{Q} \lambda \boldsymbol{x}. \boldsymbol{Q}\left(\lambda \boldsymbol{y} \lambda i. \mathit{find}\left([\iota z. \boldsymbol{y} = (\lambda j. E(z, j))], [\iota x. \boldsymbol{x} = (\lambda k. E(x, k))], i\right)\right)$
2. $[_{\mathrm{TV}}\mathsf{eats}] \rightsquigarrow \lambda \boldsymbol{Q} \lambda \boldsymbol{x}. \boldsymbol{Q}(\lambda \boldsymbol{y}. \boldsymbol{eat}(\boldsymbol{y}, \boldsymbol{x}))$
   $= \lambda \boldsymbol{Q} \lambda \boldsymbol{x}. \boldsymbol{Q}\left(\lambda \boldsymbol{y} \lambda i. \mathit{eat}\left([\iota z. \boldsymbol{y} = (\lambda j. E(z, j))], [\iota x. \boldsymbol{x} = (\lambda k. E(x, k))], i\right)\right)$
3. $[_{\mathrm{CONJ}}\mathsf{and}] \rightsquigarrow \lambda \boldsymbol{L}_1 \lambda \boldsymbol{L} \lambda \boldsymbol{Q} \lambda \boldsymbol{x} \lambda i. \boldsymbol{L}_1(\boldsymbol{Q}, \boldsymbol{x}, i) \wedge \boldsymbol{L}(\boldsymbol{Q}, \boldsymbol{x}, i)$
4. $[_{\mathrm{TV}}\mathsf{finds}][_{\mathrm{CONJ}}\mathsf{and}] \rightsquigarrow \lambda \boldsymbol{L} \lambda \boldsymbol{Q} \lambda \boldsymbol{x}. \boldsymbol{Q}(\lambda \boldsymbol{y}. \boldsymbol{find}(\boldsymbol{y}, \boldsymbol{x})) \wedge \boldsymbol{L}(\boldsymbol{Q}, \boldsymbol{x})$
   $= \lambda \boldsymbol{L}_1 \lambda \boldsymbol{L} \lambda \boldsymbol{Q} \lambda \boldsymbol{x} \lambda i. \boldsymbol{L}_1(\boldsymbol{Q}, \boldsymbol{x}, i) \wedge \boldsymbol{L}(\boldsymbol{Q}, \boldsymbol{x}, i)$
   $\left[\lambda \boldsymbol{Q}_1 \lambda \boldsymbol{z}. \boldsymbol{Q}_1\left(\lambda \boldsymbol{y} \lambda k. \mathit{find}\left([\iota z. \boldsymbol{y} = (\lambda j. E(z, j))], [\iota x. \boldsymbol{z} = (\lambda k_1. E(x, k_1))], k\right)\right)\right]$
   $= \lambda \boldsymbol{L} \lambda \boldsymbol{Q} \lambda \boldsymbol{x} \lambda i. \left[\lambda \boldsymbol{Q}_1 \lambda \boldsymbol{z}. \boldsymbol{Q}_1\left(\lambda \boldsymbol{y} \lambda k. \mathit{find}\left([\iota z. \boldsymbol{y} = (\lambda j. E(z, j))], [\iota x. \boldsymbol{z} = (\lambda k_1. E(x, k_1))], k\right)\right)\right](\boldsymbol{Q}, \boldsymbol{x}, i) \wedge \boldsymbol{L}(\boldsymbol{Q}, \boldsymbol{x}, i)$
   $= \lambda \boldsymbol{L} \lambda \boldsymbol{Q} \lambda \boldsymbol{x} \lambda i. \boldsymbol{Q}\left(\lambda \boldsymbol{y} \lambda k. \mathit{find}\left([\iota z. \boldsymbol{y} = (\lambda j. E(z, j))], [\iota x. \boldsymbol{x} = (\lambda k_1. E(x, k_1))], k\right)\right)(i) \wedge \boldsymbol{L}(\boldsymbol{Q}, \boldsymbol{x}, i)$
5. $[_{\mathrm{TV}}[_{\mathrm{TV}}\mathsf{finds}][_{\mathrm{CONJ}}\mathsf{and}][_{\mathrm{TV}}\mathsf{eats}]] \rightsquigarrow \lambda \boldsymbol{Q} \lambda \boldsymbol{x}. \boldsymbol{Q}(\lambda \boldsymbol{y}. (\boldsymbol{find} \wedge \boldsymbol{eat})(\boldsymbol{y}, \boldsymbol{x}))$
   $= \lambda \boldsymbol{L} \lambda \boldsymbol{Q} \lambda \boldsymbol{x} \lambda i. \boldsymbol{Q}\left(\lambda \boldsymbol{y} \lambda k. \mathit{find}\left([\iota z. \boldsymbol{y} = (\lambda j. E(z, j))], [\iota x. \boldsymbol{x} = (\lambda k_1. E(x, k_1))], k\right)\right)(i) \wedge \boldsymbol{L}(\boldsymbol{Q}, \boldsymbol{x}, i)$
   $\left[\lambda \boldsymbol{Q}_1 \lambda \boldsymbol{x}_1. \boldsymbol{Q}_1\left(\lambda \boldsymbol{z} \lambda k_2. \mathit{eat}\left([\iota z. \boldsymbol{z} = (\lambda k_3. E(z, k_3))], [\iota x_1. \boldsymbol{x}_1 = (\lambda k_4. E(x_1, k_4))], k_2\right)\right)\right]$
   $= \lambda \boldsymbol{Q} \lambda \boldsymbol{x} \lambda i. \boldsymbol{Q}\left(\lambda \boldsymbol{y} \lambda k. \mathit{find}\left([\iota z. \boldsymbol{y} = (\lambda j. E(z, j))], [\iota x. \boldsymbol{x} = (\lambda k_1. E(x, k_1))], k\right)\right)(i) \wedge$
   $\left[\lambda \boldsymbol{Q}_1 \lambda \boldsymbol{x}_1. \boldsymbol{Q}_1\left(\lambda \boldsymbol{z} \lambda k_2. \mathit{eat}\left([\iota z. \boldsymbol{z} = (\lambda k_3. E(z, k_3))], [\iota x_1. \boldsymbol{x}_1 = (\lambda k_4. E(x_1, k_4))], k_2\right)\right)\right](\boldsymbol{Q}, \boldsymbol{x}, i)$
   $= \lambda \boldsymbol{Q} \lambda \boldsymbol{x} \lambda i. \boldsymbol{Q}\left(\lambda \boldsymbol{y} \lambda k. \mathit{find}\left([\iota z. \boldsymbol{y} = (\lambda j. E(z, j))], [\iota x. \boldsymbol{x} = (\lambda k_1. E(x, k_1))], k\right)\right)(i) \wedge$
   $\boldsymbol{Q}\left(\lambda \boldsymbol{z} \lambda k_2. \mathit{eat}\left([\iota z. \boldsymbol{z} = (\lambda k_3. E(z, k_3))], [\iota x. \boldsymbol{x} = (\lambda k_4. E(x, k_4))], k_2\right)\right)(i)$
   $= \lambda \boldsymbol{Q} \lambda \boldsymbol{x} \lambda i. \boldsymbol{Q}\left(\lambda \boldsymbol{y}. (\mathit{find} \wedge \mathit{eat})\left([\iota y. \boldsymbol{y} = (\lambda j. E(y, j))], [\iota x. \boldsymbol{x} = (\lambda k_1. E(x, k_1))], i\right)\right)$

6. $[_{\text{NP}}[_{\text{DET}}\mathsf{a}][_{\text{N}}\mathsf{unicorn}]] \rightsquigarrow \lambda \boldsymbol{P} \bigvee \boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{P}(\boldsymbol{x})$

   $= \lambda \boldsymbol{P} \lambda i \exists \boldsymbol{x}.\, unicorn\big([\iota x.\boldsymbol{x} = (\lambda j.E(x,j))], i\big) \wedge \boldsymbol{P}(\boldsymbol{x}, i)$

7. $[_{\text{VP}}[_{\text{TV}}[_{\text{TV}}\mathsf{finds}][_{\text{CONJ}}\mathsf{and}][_{\text{TV}}\mathsf{eats}]][_{\text{NP}}[_{\text{DET}}\mathsf{a}][_{\text{N}}\mathsf{unicorn}]]] \rightsquigarrow \lambda \boldsymbol{y} \bigvee \boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge (\boldsymbol{find} \wedge \boldsymbol{eat})(\boldsymbol{x}, \boldsymbol{y})$

   $= \lambda \boldsymbol{Q} \lambda \boldsymbol{z} \lambda i.\, \boldsymbol{Q}\big(\lambda \boldsymbol{y}.(find \wedge eat)\big([\iota y.\boldsymbol{y} = (\lambda j.E(y,j))], [\iota z.\boldsymbol{z} = (\lambda k_1.E(z,k_1))], i\big)\big)$
   $\big[\lambda \boldsymbol{P} \lambda k_2 \exists \boldsymbol{x}.\, unicorn\big([\iota x.\boldsymbol{x} = (\lambda k_3.E(x,k_3))], k_2\big) \wedge \boldsymbol{P}(\boldsymbol{x}, k_2)\big]$

   $= \lambda \boldsymbol{z} \lambda i.\big[\lambda \boldsymbol{P} \lambda k_2 \exists \boldsymbol{x}.\, unicorn\big([\iota x.\boldsymbol{x} = (\lambda k_3.E(x,k_3))], k_2\big) \wedge$
   $\boldsymbol{P}(\boldsymbol{x}, k_2)\big]\big(\lambda \boldsymbol{y}.(find \wedge eat)\big([\iota y.\boldsymbol{y} = (\lambda j.E(y,j))], [\iota z.\boldsymbol{z} = (\lambda k_1.E(z,k_1))], i\big)\big)$

   $= \lambda \boldsymbol{z} \lambda i \exists \boldsymbol{x}.\, unicorn\big([\iota x.\boldsymbol{x} = (\lambda j.E(x,j))], i\big) \wedge$
   $\big(\lambda \boldsymbol{y} \lambda k.(find \wedge eat)\big([\iota y.\boldsymbol{y} = (\lambda j.E(y,j))], [\iota z.\boldsymbol{z} = (\lambda k_1.E(z,k_1))], k\big)\big)(\boldsymbol{x}, i)$

   $= \lambda \boldsymbol{z} \lambda i \exists \boldsymbol{x}.\, unicorn\big([\iota x.\boldsymbol{x} = (\lambda j.E(x,j))], i\big) \wedge (find \wedge eat)\big([\iota x.\boldsymbol{x} = (\lambda j.E(x,j))], [\iota z.\boldsymbol{z} = (\lambda k.E(z,k))], i\big)$

   $= \lambda \boldsymbol{y} \lambda i \exists x.\, unicorn(x,i) \wedge (find \wedge eat)\big(x, [\iota y.\boldsymbol{y} = (\lambda j.E(y,j))], i\big)$

8. $[_{\text{S}}[_{\text{NP}}\mathsf{John}][_{\text{VP}}[_{\text{TV}}[_{\text{TV}}\mathsf{finds}][_{\text{CONJ}}\mathsf{and}][_{\text{TV}}\mathsf{eats}]][_{\text{NP}}[_{\text{DET}}\mathsf{a}][_{\text{N}}\mathsf{unicorn}]]]] \rightsquigarrow \bigvee \boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge (\boldsymbol{find} \wedge \boldsymbol{eat})(\boldsymbol{x}, \boldsymbol{john})$

   $= \lambda \boldsymbol{y} \lambda i \exists x.\, unicorn(x,i) \wedge (find \wedge eat)\big(x, [\iota y.\boldsymbol{y} = (\lambda j.E(y,j))], i\big)\,[\lambda k.E(john,k)]$

   $= \lambda i \exists x.\, unicorn(x,i) \wedge (find \wedge eat)\big(x, [\iota y.[\lambda k.E(john,k)] = (\lambda j.E(y,j))], i\big)$

   $= \lambda i \exists x.\, unicorn(x,i) \wedge (find \wedge eat)(x, john, i)$

The WTY_1^3 translation of the logical form of the sentence Bill is a man (cf. (3.2.20)) is defined as follows:

(7.2.24)

1. $[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{man}]] \rightsquigarrow \lambda\boldsymbol{P}\bigvee\boldsymbol{x}.\boldsymbol{man}(\boldsymbol{x}) \wedge \boldsymbol{P}(\boldsymbol{x})$
   $= \lambda\boldsymbol{P}\lambda i\exists\boldsymbol{x}.man\left([\iota x.\boldsymbol{x} = (\lambda j.E(x,j))],i\right) \wedge \boldsymbol{P}(\boldsymbol{x},i)$
2. $[\mathsf{is}] \rightsquigarrow \lambda\boldsymbol{Q}\lambda\boldsymbol{x}.\boldsymbol{Q}(\lambda\boldsymbol{y}.\boldsymbol{x} \doteq \boldsymbol{y}) = \lambda\boldsymbol{Q}\lambda\boldsymbol{x}.\boldsymbol{Q}(\lambda\boldsymbol{y}\lambda i.\boldsymbol{x}(i) = \boldsymbol{y}(i))$
3. $[_{\mathrm{VP}}[\mathsf{is}][_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{man}]]] \rightsquigarrow \lambda\boldsymbol{y}\bigvee\boldsymbol{x}.\boldsymbol{man}(\boldsymbol{x}) \wedge \boldsymbol{y} \doteq \boldsymbol{x}$
   $= \lambda\boldsymbol{Q}\lambda\boldsymbol{z}.\boldsymbol{Q}(\lambda\boldsymbol{y}\lambda k.\boldsymbol{z}(k) = \boldsymbol{y}(k))$
   $\left[\lambda\boldsymbol{P}\lambda i\exists\boldsymbol{x}.man\left([\iota x.\boldsymbol{x} = (\lambda j.E(x,j))],i\right) \wedge \boldsymbol{P}(\boldsymbol{x},i)\right]$
   $= \lambda\boldsymbol{z}.\left[\lambda\boldsymbol{P}\lambda i\exists\boldsymbol{x}.man\left([\iota x.\boldsymbol{x} = (\lambda j.E(x,j))],i\right) \wedge \boldsymbol{P}(\boldsymbol{x},i)\right](\lambda\boldsymbol{y}\lambda k.\boldsymbol{z}(k) = \boldsymbol{y}(k))$
   $= \lambda\boldsymbol{z}\lambda i\exists\boldsymbol{x}.man\left([\iota x.\boldsymbol{x} = (\lambda j.E(x,j))],i\right) \wedge (\lambda\boldsymbol{y}\lambda k.\boldsymbol{z}(k) = \boldsymbol{y}(k))(\boldsymbol{x},i)$
   $= \lambda\boldsymbol{z}\lambda i\exists\boldsymbol{x}.man\left([\iota x.\boldsymbol{x} = (\lambda j.E(x,j))],i\right) \wedge \boldsymbol{z}(i) = \boldsymbol{x}(i)$
4. $[_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{Bill}][_{\mathrm{VP}}[\mathsf{is}][_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{man}]]]] \rightsquigarrow \boldsymbol{man}(\boldsymbol{bill})$
   $= \lambda\boldsymbol{z}\lambda i\exists\boldsymbol{x}.man\left([\iota x.\boldsymbol{x} = (\lambda j.E(x,j))],i\right) \wedge \boldsymbol{z}(i) = \boldsymbol{x}(i)\,[\lambda k.E(bill,k)]$
   $= \lambda i\exists\boldsymbol{x}.man\left([\iota x.\boldsymbol{x} = (\lambda j.E(x,j))],i\right) \wedge [\lambda k.E(bill,k)](i) = \boldsymbol{x}(i)$
   $= \lambda i\exists x.man(x,i) \wedge E(bill,i) = E(x,i)$
   $= \lambda i\exists x.man(x,i) \wedge bill = x$
   $= \lambda i.man(bill,i)$

D.3. 'Strong' PTQ Definitions

This section supplies (the derivations of) some definitions of the STY_1^3 translations of the logical sentence forms from Chapter 8.2.2. In particular, the remaining definitions of the forms from (3.2.2) to (3.2.27) are given below:

(D.3.1) $[_{\mathrm{S}}[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{every}][_{\mathrm{N}}\mathsf{man}]][_{\mathrm{VP}}[_{\mathrm{IV}}\mathsf{walks}]]] \rightsquigarrow \bigwedge \boldsymbol{x}.\boldsymbol{man}(\boldsymbol{x}) \dot{\rightarrow} \boldsymbol{walk}(\boldsymbol{x})$

$= \lambda i\lambda j\forall x.\big(man(x,j) \rightarrow walk(x,j)\big) \wedge \big(\forall p.\big(p(i) \wedge \big(\exists y.\, abt(y,p) \wedge abt\big(y,[\lambda k.\, man(x,k) \rightarrow walk(x,k)]\big)\big) \twoheadrightarrow p(j)\big)\big)$

(D.3.2) $[_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{John}][_{\mathrm{VP}}[_{\mathrm{IV}}\mathsf{finds}][_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{unicorn}]]]]$

$\rightsquigarrow \bigvee \boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{find}(\boldsymbol{x},\boldsymbol{john})$

$= \lambda i\lambda j\exists x.\big(unicorn(x,j) \wedge find\big([\lambda P\lambda k.\, P(x,k)], john, j\big)\big) \wedge \big(\forall p.(p(i) \wedge (abt(x,p) \vee abt(john,p))) \twoheadrightarrow p(j)\big)$

(D.3.3) $[_{\mathrm{S}}[[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{price}]][_{\mathrm{VP}}[\mathsf{rises}]]] \rightsquigarrow \bigvee \boldsymbol{P}.\boldsymbol{price}(\boldsymbol{P}) \wedge \boldsymbol{rise}(\boldsymbol{P})$

$= \lambda i\lambda j\exists T.(price(T,j) \wedge rise(T,j)) \wedge \big(\forall p.\big(p(i) \wedge \big(\exists x.\, abt(x,p) \wedge abt\big(x,[\lambda k.\, price(T,k) \wedge rise(T,k)]\big)\big)\big) \twoheadrightarrow p(j)\big)$

(D.3.4) $[_{\mathrm{S}}[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{the}][_{\mathrm{N}}\mathsf{temperature}]][_{\mathrm{IV}}\mathsf{rises}]]$

$\rightsquigarrow \bigvee \boldsymbol{P} \bigwedge \boldsymbol{P}_1.(\boldsymbol{temp}(\boldsymbol{P}_1) \dot{\leftrightarrow} \boldsymbol{P} \doteq \boldsymbol{P}_1) \wedge \boldsymbol{rise}(\boldsymbol{P})$

$= \lambda i\lambda j\exists T\forall T_1.((temp(T_1,j) \leftrightarrow T = T_1) \wedge rise(T,j)) \wedge \big(\forall p.\big(p(i) \wedge \big(\exists x.\, abt(x,p) \wedge abt\big(x,[\lambda k.(temp(T_1,k) \leftrightarrow T = T_1) \wedge rise(T,k)]\big)\big)\big) \twoheadrightarrow p(j)\big)$

(D.3.5) $[_{\mathrm{S}}[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{woman}]][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{loves}][_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{every}][_{\mathrm{N}}\mathsf{man}]]]]$

$\rightsquigarrow \bigvee \boldsymbol{x}.\boldsymbol{woman}(\boldsymbol{x}) \wedge \big(\bigwedge \boldsymbol{y}.\boldsymbol{man}(\boldsymbol{y}) \dot{\rightarrow} \boldsymbol{love}(\boldsymbol{y},\boldsymbol{x})\big)$

$= \lambda i\lambda j\exists x.\big(woman(x,j) \wedge (\forall y.\, man(y,j) \rightarrow love(y,x,j))\big) \wedge \big(\forall p.\big(p(i) \wedge \big(\exists z.\, abt\big(z,[\lambda k.woman(x,k) \wedge (\forall y.\, man(y,k) \rightarrow love(y,x,k))]\big) \wedge abt(z,p)\big)\big) \twoheadrightarrow p(j)\big)$

(D.3.6) $[[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{every}][_{\mathrm{N}}\mathsf{man}]]^1[_{\mathrm{S}}[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{woman}]][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{loves}]\ t_1]]]$

$\rightsquigarrow \bigwedge \boldsymbol{y}.\boldsymbol{man}(\boldsymbol{y}) \dot{\rightarrow} \big(\bigvee \boldsymbol{x}.\boldsymbol{woman}(\boldsymbol{x}) \wedge \boldsymbol{love}(\boldsymbol{y},\boldsymbol{x})\big)$

$$= \lambda i\lambda j\forall y.\big(man\,(y,j) \rightarrow \big(\exists x.\,woman\,(x,j) \wedge love\,(y,x,j)\big)\big) \wedge$$
$$\big(\forall p.\big(p(i) \wedge \big(\exists z.\,abt\big(z,[\lambda k.\,man\,(y,k)\ \rightarrow$$
$$(\exists x.\,woman\,(x,k) \wedge love\,(y,x,k))]\big) \wedge abt(z,p)\big)\big) \twoheadrightarrow p(j)\big)$$

(D.3.7) $[_{S}[_{NP}\mathsf{Bill}][_{VP}[\mathsf{is}][_{NP}[_{DET}\mathsf{a}][_{N}\mathsf{man}]]]] \rightsquigarrow \boldsymbol{man}\,(\boldsymbol{bill})$

$$= \lambda i\lambda j.\,man\,(bill,j) \wedge (\forall p.(p(i) \wedge abt\,(bill,p)) \twoheadrightarrow p(j))$$

(D.3.8) $[_{S}[_{NP}[_{DET}\mathsf{the}][_{N}\mathsf{temperature}]][_{VP}[_{TV}\mathsf{is}][_{NP}\mathsf{ninety}]]]$

$$\rightsquigarrow \bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}.(\boldsymbol{temp}\,(\lambda \boldsymbol{x}_1.\,\boldsymbol{y}) \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}) \wedge \boldsymbol{x} \doteq \boldsymbol{ninety}$$
$$= \lambda i\lambda j\exists x\forall y.\,\big(\big(\big(\exists T.\,temp\,(T,i) \wedge y = T\,(\boldsymbol{w})\big) \leftrightarrow x = y\big) \wedge x = ninety\big) \wedge$$
$$\big(\forall p.\big(p(i) \wedge \big(\exists y.\,abt\,(y,p) \wedge abt\,\big(y,[\lambda k.\,\big(\big(\exists T.\,temp\,(T,k) \wedge$$
$$y = T\,(\boldsymbol{w})\big) \leftrightarrow x = y\big) \wedge x = ninety]\big)\big) \twoheadrightarrow p(j)\big)\big)$$

The definition of the STY_1^3 translation of the LF of the sentence The man walks (cf. (3.2.4)) is obtained as follows:
(8.2.9)

1. $[_{\mathrm{DET}}\mathsf{the}] \rightsquigarrow \lambda\boldsymbol{P}_1\lambda\boldsymbol{P}\bigvee\boldsymbol{x}\bigwedge\boldsymbol{y}.(\boldsymbol{P}_1(\boldsymbol{y}) \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}) \wedge \boldsymbol{P}(\boldsymbol{x})$
   $= \lambda\boldsymbol{P}_1\lambda\boldsymbol{P}\lambda i\lambda j\exists\boldsymbol{x}\forall\boldsymbol{y}.(\boldsymbol{P}_1(\boldsymbol{y},i,j) \leftrightarrow \boldsymbol{x}(i,j) = \boldsymbol{y}(i,j)) \wedge \boldsymbol{P}(\boldsymbol{x},i,j)$
2. $[_{\mathrm{N}}\mathsf{man}] \rightsquigarrow \boldsymbol{man} = \lambda\boldsymbol{x}\lambda i\lambda j.man(\boldsymbol{x}^\circ,j) \wedge \big(\forall p.(p(i) \wedge (\exists y.abt(y,[\lambda k.man(\boldsymbol{x}^\circ,k)]) \wedge abt(y,p))) \twoheadrightarrow p(j)\big)$
3. $[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{the}][_{\mathrm{N}}\mathsf{man}]] \rightsquigarrow \lambda\boldsymbol{P}\bigvee\boldsymbol{x}\bigwedge\boldsymbol{y}.(\boldsymbol{man}(\boldsymbol{y}) \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}) \wedge \boldsymbol{P}(\boldsymbol{x})$
   $= \lambda\boldsymbol{P}_1\lambda\boldsymbol{P}\lambda i\lambda j\exists\boldsymbol{x}\forall\boldsymbol{y}.(\boldsymbol{P}_1(\boldsymbol{y},i,j) \leftrightarrow \boldsymbol{x}(i,j) = \boldsymbol{y}(i,j)) \wedge \boldsymbol{P}(\boldsymbol{x},i,j)[\lambda\boldsymbol{z}\lambda k\lambda k_1.man(\boldsymbol{z}^\circ,k_1) \wedge$
   $\big(\forall p.(p(k) \wedge (\exists z.abt(z,[\lambda k_2.man(\boldsymbol{z}^\circ,k_2)]) \wedge abt(z,p))) \twoheadrightarrow p(k_1)\big)]$
   $= \lambda\boldsymbol{P}\lambda i\lambda j\exists\boldsymbol{x}\forall\boldsymbol{y}.([\lambda\boldsymbol{z}\lambda k\lambda k_1.man(\boldsymbol{z}^\circ,k_1) \wedge \big(\forall p.(p(k) \wedge (\exists z.abt(z,[\lambda k_2.man(\boldsymbol{z}^\circ,k_2)]) \wedge$
   $abt(z,p))) \twoheadrightarrow p(k_1)\big)](\boldsymbol{y},i,j) \leftrightarrow \boldsymbol{x}(i,j) = \boldsymbol{y}(i,j)) \wedge \boldsymbol{P}(\boldsymbol{x},i,j)$
   $= \lambda\boldsymbol{P}\lambda i\lambda j\exists\boldsymbol{x}\forall\boldsymbol{y}.\big(man(\boldsymbol{y}^\circ,j) \wedge \big(\forall p.(p(i) \wedge (\exists y.abt(y,[\lambda k.man(\boldsymbol{y}^\circ,j)]) \wedge abt(y,p))) \twoheadrightarrow p(i)\big)\big) \leftrightarrow$
   $\boldsymbol{x}(i,j) = \boldsymbol{y}(i,j)) \wedge \boldsymbol{P}(\boldsymbol{x},i,j)$
4. $[_{\mathrm{VP}}[_{\mathrm{IV}}\mathsf{walks}]] \rightsquigarrow \boldsymbol{walk} = \lambda\boldsymbol{x}\lambda i\lambda j.walk(\boldsymbol{x}^\circ,j) \wedge \big(\forall p.(p(i) \wedge (\exists y.abt(y,[\lambda k.walk(\boldsymbol{x}^\circ,k)]) \wedge abt(y,p))) \twoheadrightarrow p(j)\big)$
5. $[_{\mathrm{S}}[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{the}][_{\mathrm{N}}\mathsf{man}]][_{\mathrm{VP}}[_{\mathrm{IV}}\mathsf{walks}]]] \rightsquigarrow \bigvee\boldsymbol{x}\bigwedge\boldsymbol{y}.(\boldsymbol{man}(\boldsymbol{y}) \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}) \wedge \boldsymbol{walk}(\boldsymbol{x})$
   $= \lambda\boldsymbol{P}\lambda i\lambda j\exists\boldsymbol{x}\forall\boldsymbol{y}.\big(man(\boldsymbol{y}^\circ,j) \wedge \big(\forall p.(p(i) \wedge (\exists y.abt(y,[\lambda k.man(\boldsymbol{y}^\circ,j)]) \wedge abt(y,p))) \twoheadrightarrow p(i)\big)\big) \leftrightarrow$
   $\boldsymbol{x}(i,j) = \boldsymbol{y}(i,j)\big) \wedge \boldsymbol{P}(\boldsymbol{x},i,j)[\lambda\boldsymbol{z}\lambda k_1\lambda k_2.walk(\boldsymbol{z}^\circ,k_2) \wedge$
   $\big(\forall q.(q(k_1) \wedge (\exists x_1.abt(x_1,[\lambda k_3.walk(\boldsymbol{z}^\circ,k_3)]) \wedge abt(x_1,q))) \twoheadrightarrow q(k_2)\big)]$
   $= \lambda i\lambda j\exists\boldsymbol{x}\forall\boldsymbol{y}.\big(man(\boldsymbol{y}^\circ,j) \wedge \big(\forall p.(p(i) \wedge (\exists y.abt(y,[\lambda k.man(\boldsymbol{y}^\circ,j)]) \wedge abt(y,p))) \twoheadrightarrow p(i)\big)\big) \leftrightarrow \boldsymbol{x}(i,j) = \boldsymbol{y}(i,j)) \wedge$
   $\big(walk(\boldsymbol{x}^\circ,j) \wedge \big(\forall q.(q(i) \wedge (\exists z.abt(z,[\lambda k_1.walk(\boldsymbol{x}^\circ,k_1)]) \wedge abt(z,q))) \twoheadrightarrow q(j)\big)\big)$
   $= \lambda i\lambda j\exists x.\big(\forall y.(man(y,j) \leftrightarrow x = y) \wedge walk(x,j)\big) \wedge \big(\forall p.\big(p(i) \wedge$
   $\big(\exists z.abt(z,p) \wedge abt\big(z,\big[\lambda k.(\forall x_1.(man(x_1,k) \leftrightarrow x = x_1) \wedge walk(x,k))\big]\big)\big) \twoheadrightarrow p(j)\big)\big)$

The STY_1^3 translation of the logical form of the sentence John finds Mary has the following definition:

(8.2.10)

1. $[_{\text{NP}}\text{Mary}] \rightsquigarrow \boldsymbol{mary} = \lambda i \lambda j \forall p.(p(i) \wedge abt\,(mary, p)) \twoheadrightarrow p(j)$
2. $[_{\text{TV}}\text{finds}] \rightsquigarrow \lambda \boldsymbol{Q} \lambda \boldsymbol{x}.\boldsymbol{Q}(\lambda \boldsymbol{y}.\boldsymbol{find}(\boldsymbol{y}, \boldsymbol{x}))$
   $= \lambda \boldsymbol{Q} \lambda \boldsymbol{x}.\boldsymbol{Q}\big(\lambda \boldsymbol{y} \lambda i \lambda j.find(\boldsymbol{y}^\circ, \boldsymbol{x}^\circ, j) \wedge$
   $\big(\forall p.(p(i) \wedge (\exists z.abt\,(z, [\lambda k.find\,(\boldsymbol{y}^\circ, \boldsymbol{x}^\circ, k)]) \wedge abt\,(z, p))) \twoheadrightarrow p(j)\big)\big)$
3. $[_{\text{VP}}[_{\text{TV}}\text{finds}][_{\text{NP}}\text{Mary}]] \rightsquigarrow \lambda \boldsymbol{x}.\boldsymbol{find}(\boldsymbol{mary}, \boldsymbol{x})$
   $= \lambda \boldsymbol{Q} \lambda \boldsymbol{x}.\boldsymbol{Q}\big(\lambda \boldsymbol{y} \lambda i \lambda j.find(\boldsymbol{y}^\circ, \boldsymbol{x}^\circ, j) \wedge \big(\forall p.(p(i) \wedge (\exists z.abt\,(z, [\lambda k.find\,(\boldsymbol{y}^\circ, \boldsymbol{x}^\circ, k)]) \wedge abt\,(z, p))) \twoheadrightarrow p(j)\big)\big)$
   $\underline{\big[\lambda \boldsymbol{P}.\boldsymbol{P}\big(\lambda k \lambda k_1 \big(\forall q.\big(q(k) \wedge abt\,(mary, q)\big) \twoheadrightarrow q(k_1)\big)\big]}$
   $= \lambda \boldsymbol{x}.\underline{\big[\lambda \boldsymbol{P}.\boldsymbol{P}\big(\lambda k \lambda k_1 \forall q.\big(q(k) \wedge abt\,(mary, q)\big) \twoheadrightarrow q(k_1)\big)\big]}$
   $\big(\lambda \boldsymbol{y} \lambda i \lambda j.find(\boldsymbol{y}^\circ, \boldsymbol{x}^\circ, j) \wedge \big(\forall p.(p(i) \wedge (\exists z.abt\,(z, [\lambda k.find\,(\boldsymbol{y}^\circ, \boldsymbol{x}^\circ, k)]) \wedge abt\,(z, p))) \twoheadrightarrow p(j)\big)\big)$
   $= \lambda \boldsymbol{x}.\big(\lambda \boldsymbol{y} \lambda i \lambda j.find(\boldsymbol{y}^\circ, \boldsymbol{x}^\circ, j) \wedge \big(\forall p.(p(i) \wedge (\exists z.abt\,(z, [\lambda k.find\,(\boldsymbol{y}^\circ, \boldsymbol{x}^\circ, k)]) \wedge abt\,(z, p))) \twoheadrightarrow p(j)\big)\big)$
   $\underline{\big(\lambda k \lambda k_1 \forall q.\big(q(k) \wedge abt\,(mary, q) \twoheadrightarrow q(k_1)\big)\big)}$
   $= \lambda \boldsymbol{x} \lambda i \lambda j.find\big(\underline{\big(\lambda k \lambda k_1 \forall q.\big(q(k) \wedge abt\,(mary, q)\big) \twoheadrightarrow q(k_1)\big)}^\circ, \boldsymbol{x}^\circ, j\big) \wedge \big(\forall p.\big(p(i) \wedge \big(\exists z.abt\,\big(z, \big[\lambda k.find$
   $\underline{\big(\big(\lambda k \lambda k_1 \forall q.\big(q(k) \wedge abt\,(mary, q) \twoheadrightarrow q(k_1)\big)\big)^\circ}, \boldsymbol{x}^\circ, k\big)\big]\big) \wedge abt\,(z, p)\big)\big)\big) \twoheadrightarrow p(j)\big)$
   $= \lambda \boldsymbol{x} \lambda i \lambda j.find\,(mary, \boldsymbol{x}^\circ, j) \wedge \big(\forall p.(p(i) \wedge (\exists z.abt\,(z, [\lambda k.find\,(mary, \boldsymbol{x}^\circ, k)]) \wedge abt\,(z, p)))) \twoheadrightarrow p(j)\big)$
4. $[_{\text{NP}}\text{John}] \rightsquigarrow \boldsymbol{john} = \lambda i \lambda j \forall p.\big(p(i) \wedge abt\,(john, p)\big) \twoheadrightarrow p(j)$
5. $[_{\text{S}}[_{\text{NP}}\text{John}][_{\text{VP}}[_{\text{TV}}\text{finds}][_{\text{NP}}\text{Mary}]]] \rightsquigarrow \boldsymbol{find}(\boldsymbol{mary}, \boldsymbol{john})$
   $= \lambda \boldsymbol{x} \lambda i \lambda j.find\,(mary, \boldsymbol{x}^\circ, j) \wedge \big(\forall p.(p(i) \wedge (\exists z.abt\,(z, [\lambda k.find\,(mary, \boldsymbol{x}^\circ, k)]) \wedge abt\,(z, p))) \twoheadrightarrow p(j)\big)$
   $\underline{\big[\lambda \boldsymbol{P}.\boldsymbol{P}\big(\lambda k \lambda k_1 \forall q.\big(q(k) \wedge abt\,(john, q)\big) \twoheadrightarrow q(k_1)\big)\big]}$
   $= \lambda i \lambda j.find\,(mary, john, j) \wedge \big(\forall p.(p(i) \wedge (\exists z.abt\,(z, [\lambda k.find\,(mary, john, k)] \wedge abt\,(z, p)))) \twoheadrightarrow p(j)\big)$

The definition of the STY_1^3 translation of the narrow-scope reading of the sentence John seeks a unicorn (cf. (3.2.13)) is given below. In the derivation, we abbreviate $\lambda P \lambda k \exists x.\, unicorn(x,k)$ as 'Z':

(8.2.12)

1. $[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{unicorn}]] \rightsquigarrow \lambda \boldsymbol{P} \bigvee \boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{P}(\boldsymbol{x})$
   $= \lambda \boldsymbol{P} \lambda i \lambda j \exists \boldsymbol{x}.\big(unicorn(\boldsymbol{x}^\circ, j) \wedge \big(\forall p.(p(i) \wedge (\exists y.\, abt(y, [\lambda k.\, unicorn(\boldsymbol{x}^\circ, j)]) \wedge abt(y,p))) \twoheadrightarrow p(i)\big)\big) \wedge \boldsymbol{P}(\boldsymbol{x}, i, j)$
2. $[_{\mathrm{TV}}\mathsf{seeks}] \rightsquigarrow \boldsymbol{seek} = \lambda \boldsymbol{Q} \lambda \boldsymbol{x} \lambda i \lambda j.\, seek(\boldsymbol{Q}^\circ, \boldsymbol{x}^\circ, j) \wedge \big(\forall p.(p(i) \wedge (\exists z.(abt(z, [\lambda k.\, seek(\boldsymbol{Q}^\circ, \boldsymbol{x}^\circ, k)]) \wedge abt(z,p)))) \twoheadrightarrow p(j)\big)$
3. $[_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{seeks}][_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{unicorn}]]] \rightsquigarrow \lambda \boldsymbol{y}.\boldsymbol{seek}\big([\lambda \boldsymbol{P} \bigvee \boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{P}(\boldsymbol{x})], \boldsymbol{y}\big)$
   $= \lambda \boldsymbol{Q} \lambda \boldsymbol{y} \lambda i \lambda j.\, seek(\boldsymbol{Q}^\circ, \boldsymbol{y}^\circ, j) \wedge \big(\forall p.(p(i) \wedge (\exists z.\, abt(z, [\lambda k.\, seek(\boldsymbol{Q}^\circ, \boldsymbol{y}^\circ, k)]) \wedge abt(z,p))) \twoheadrightarrow p(j)\big)\big[\lambda \boldsymbol{P} \lambda k_2 \lambda k_1 \exists \boldsymbol{x}.$
   $\big(unicorn(\boldsymbol{x}^\circ, k_1) \wedge \big(\forall q.(q(k_2) \wedge (\exists y.\, abt(y, [\lambda k_3.\, unicorn(\boldsymbol{x}^\circ, k_3)]) \wedge abt(y,q))) \twoheadrightarrow q(k_2)\big)\big) \wedge \boldsymbol{P}(\boldsymbol{x}, k_2, k_1)\big]$
   $= \lambda \boldsymbol{y} \lambda i \lambda j.\, seek(\big[\lambda \boldsymbol{P} \lambda k_2 \lambda k_1 \exists \boldsymbol{x}.\big(unic.(\boldsymbol{x}^\circ, k_1) \wedge \big(\forall q.(q(k_2) \wedge (\exists y.\, abt(y, [\lambda k_3.\, unicorn(\boldsymbol{x}^\circ, k_3)]) \wedge abt(y,q))) \twoheadrightarrow q(k_2)\big)\big) \wedge$
   $\boldsymbol{P}(\boldsymbol{x}, k_2, k_1)\big]^\circ, \boldsymbol{y}^\circ, j) \wedge \big(\forall p.(p(i) \wedge (\exists z.\, abt(z, [\lambda k.\, seek(\big[\lambda \boldsymbol{P} \lambda k_2 \lambda k_1 \exists \boldsymbol{x}.\big(unicorn(\boldsymbol{x}^\circ, k_1) \wedge \big(\forall q.(q(k_2) \wedge$
   $(\exists y.(abt(y, [\lambda k_3.\, unicorn(\boldsymbol{x}^\circ, k_3)]) \wedge abt(y,q)))) \twoheadrightarrow q(k_2)\big)\big) \wedge \boldsymbol{P}(\boldsymbol{x}, k_2, k_1)\big]^\circ, \boldsymbol{y}^\circ, k)]) \wedge abt(z,p)) \twoheadrightarrow p(j)\big)$
   $= \lambda \boldsymbol{y} \lambda i \lambda j.\, seek\big(Z, \boldsymbol{y}^\circ, j\big) \wedge \big(\forall p.(p(i) \wedge (\exists z.\, abt(z, [\lambda k_1.\, seek(Z, \boldsymbol{y}^\circ, k_1)]) \wedge abt(z,p))) \twoheadrightarrow p(j)\big)$
4. $[_{\mathrm{NP}}\mathsf{John}] \rightsquigarrow \boldsymbol{john} = \lambda i \lambda j \forall p.\big(p(i) \wedge abt(p, john)\big) \twoheadrightarrow p(j)$
5. $[_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{John}][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{seeks}][_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{unicorn}]]]] \rightsquigarrow \boldsymbol{seek}\big([\lambda \boldsymbol{P} \bigvee \boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{P}(\boldsymbol{x})], \boldsymbol{john}\big)$
   $= \lambda \boldsymbol{y} \lambda i \lambda j.\, seek\big(Z, \boldsymbol{y}^\circ, j\big) \wedge \big(\forall p.(p(i) \wedge (\exists z.\, abt(z, [\lambda k.\, seek(Z, \boldsymbol{y}^\circ, k)]) \wedge abt(z,p))) \twoheadrightarrow p(j)\big)$
   $\big[\lambda k_2 \lambda k_1 \forall q.\big(q(k_2) \wedge abt(q, john)\big) \twoheadrightarrow q(k_1)\big]$
   $= \lambda i \lambda j.\, seek\big(Z, \big[\lambda k_2 \lambda k_1 \forall q.\big(q(k_2) \wedge abt(q, john)\big) \twoheadrightarrow q(k_1)\big]^\circ, j\big) \wedge \big(\forall p.(p(i) \wedge (\exists z.\, abt(z, [\lambda k.\, seek$
   $(Z, \big[\lambda k_2 \lambda k_1.E(john, k_1) \wedge \big(\forall q.\big(q(k_2) \wedge abt(q, john)\big) \twoheadrightarrow q(k_1)\big)\big]^\circ, k)]) \wedge abt(z,p))) \twoheadrightarrow p(j)\big)$
   $= \lambda i \lambda j.\, seek\big(Z, john, j\big) \wedge \big(\forall p.\big(p(i) \wedge \big(\exists z.\, abt\big(z, \big[\lambda k_1.\, seek\big(Z, john, k_1\big)\big]\big) \wedge abt(z,p)\big)\big) \twoheadrightarrow p(j)\big)$

The definition of the STY_1^3 translation of the logical form of the sentence John does not find a unicorn (cf. (3.2.27)) runs as follows:

(8.2.14)

1. $[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{unicorn}]] \rightsquigarrow \lambda\boldsymbol{P}\bigvee\boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{P}(\boldsymbol{x})$
   $= \lambda\boldsymbol{P}\lambda i\lambda j\exists\boldsymbol{x}.\big(unicorn(\boldsymbol{x}^\circ, j) \wedge \big(\forall p.(p(i) \wedge (\exists y.\, abt(y, [\lambda k.\, unicorn(\boldsymbol{x}^\circ, j)]) \wedge abt(y, p))) \twoheadrightarrow p(i)\big)\big) \wedge \boldsymbol{P}(\boldsymbol{x}, i, j)$
2. $[_{\mathrm{TV}}\mathsf{find}] \rightsquigarrow \lambda\boldsymbol{Q}\lambda\boldsymbol{x}.\boldsymbol{Q}(\lambda\boldsymbol{y}.\boldsymbol{find}(\boldsymbol{y}, \boldsymbol{x}))$
   $= \lambda\boldsymbol{Q}\lambda\boldsymbol{x}.\boldsymbol{Q}\left(\lambda\boldsymbol{y}\lambda i\lambda j.find(\boldsymbol{y}^\circ, \boldsymbol{x}^\circ, j) \wedge \big(\forall p.(p(i) \wedge (\exists z.\, abt(z, [\lambda k.find(\boldsymbol{y}^\circ, \boldsymbol{x}^\circ, k)]) \wedge abt(z, p))) \twoheadrightarrow p(j)\big)\right)$
3. $[_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{find}][_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{unicorn}]]] \rightsquigarrow \lambda\boldsymbol{y}\bigvee\boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{find}(\boldsymbol{x}, \boldsymbol{y})$
   $= \lambda\boldsymbol{Q}\lambda\boldsymbol{x}.\boldsymbol{Q}\left(\lambda\boldsymbol{y}\lambda k_1\lambda k_2.find(\boldsymbol{y}^\circ, \boldsymbol{x}^\circ, k_2) \wedge \big(\forall p.(p(k_1) \wedge (\exists z.\, abt(z, [\lambda k.find(\boldsymbol{y}^\circ, \boldsymbol{x}^\circ, k)]) \wedge abt(z, p))) \twoheadrightarrow p(k_2)\big)\right)$
   $\big[\lambda\boldsymbol{P}\lambda i\lambda j\exists\boldsymbol{z}.\big(unicorn(\boldsymbol{z}^\circ, j) \wedge \big(\forall p.(p(i) \wedge (\exists y.\, abt(y, [\lambda k_3.\, unicorn(\boldsymbol{x}^\circ, k_3)]) \wedge abt(y, p))) \twoheadrightarrow p(j)\big)\big) \wedge \boldsymbol{P}(\boldsymbol{z}, i, j)\big]$
   $= \lambda\boldsymbol{x}.\big[\lambda\boldsymbol{P}\lambda i\lambda j\exists\boldsymbol{z}.\big(unicorn(\boldsymbol{z}^\circ, j) \wedge \big(\forall p.(p(i) \wedge (\exists y.\, abt(y, [\lambda k_3.\, unicorn(\boldsymbol{x}^\circ, k_3)]) \wedge abt(y, p))) \twoheadrightarrow p(j)\big)\big) \wedge \boldsymbol{P}(\boldsymbol{z}, i, j)\big]$
   $\big(\lambda\boldsymbol{y}\lambda k_1\lambda k_2.find(\boldsymbol{y}^\circ, \boldsymbol{x}^\circ, k_2) \wedge \big(\forall p.(p(k_1) \wedge (\exists z.\, abt(z, [\lambda k.find(\boldsymbol{y}^\circ, \boldsymbol{x}^\circ, k)]) \wedge abt(z, p))) \twoheadrightarrow p(k_2)\big)\big)$
   $= \lambda\boldsymbol{x}\lambda i\lambda j\exists\boldsymbol{z}.\,\big(unicorn(\boldsymbol{z}^\circ, j) \wedge \big(\forall p.(p(i) \wedge (\exists y.\, abt(y, [\lambda k_3.\, unicorn(\boldsymbol{x}^\circ, k_3)]) \wedge abt(y, p))) \twoheadrightarrow p(j)\big)\big) \wedge$
   $\big(\lambda\boldsymbol{y}\lambda k_1\lambda k_2.find(\boldsymbol{y}^\circ, \boldsymbol{x}^\circ, k_2) \wedge \big(\forall p.(p(k_1) \wedge (\exists z.\, abt(z, [\lambda k.find(\boldsymbol{y}^\circ, \boldsymbol{x}^\circ, k)]) \wedge abt(z, p))) \twoheadrightarrow p(k_2)\big)\big)(\boldsymbol{z}, i, j)$
   $= \lambda\boldsymbol{y}\lambda i\lambda j\exists x.\big(unicorn(x, j) \wedge \big(\forall p.(p(i) \wedge (\exists z.\, abt(z, [\lambda k_3.\, unicorn(x, k_3)]) \wedge abt(z, p))) \twoheadrightarrow p(j)\big)\big) \wedge$
   $\big(find(x, \boldsymbol{y}^\circ, j) \wedge \big(\forall p.(p(i) \wedge (\exists z.\, abt(z, [\lambda k.find(x, \boldsymbol{y}^\circ, k)]) \wedge abt(z, p))) \twoheadrightarrow p(j)\big)\big)$
   $= \lambda\boldsymbol{y}\lambda i\lambda j\exists x.\big((unicorn(x, j) \wedge find(x, \boldsymbol{y}^\circ, j)) \wedge \big(\forall p.(p(i) \wedge (\exists z.\, abt(z, [\lambda k.find(x, \boldsymbol{y}^\circ, k)]) \wedge abt(z, p))) \twoheadrightarrow p(j)\big)\big)$
4. $[_{\mathrm{ADV}}\mathsf{does\ not}] \rightsquigarrow \lambda\boldsymbol{P}\lambda\boldsymbol{x}.\neg\boldsymbol{P}(\boldsymbol{x}) = \lambda\boldsymbol{P}\lambda\boldsymbol{x}\lambda i\lambda j.\neg\boldsymbol{P}(\boldsymbol{x}, i, j)$
5. $[_{\mathrm{VP}}[_{\mathrm{ADV}}\mathsf{does\ not}][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{find}][_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{unicorn}]]]] \rightsquigarrow \lambda\boldsymbol{y}.\neg\big(\bigvee\boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{find}(\boldsymbol{x}, \boldsymbol{y})\big)$
   $= \lambda\boldsymbol{P}\lambda\boldsymbol{x}\lambda i\lambda j.\neg\boldsymbol{P}(\boldsymbol{x}, i, j)\,\big[\lambda\boldsymbol{y}\lambda k_1\lambda k_2\exists x.\big((unicorn(x, k_2) \wedge find(x, \boldsymbol{y}^\circ, k_2)) \wedge$
   $\big(\forall p.(p(k_1) \wedge (\exists z.\, abt(z, [\lambda k.find(x, \boldsymbol{y}^\circ, k)]) \wedge abt(z, p))) \twoheadrightarrow p(k_2)\big)\big)\big]$

$= \lambda \boldsymbol{x} \lambda i \lambda j. \urcorner \big[\lambda \boldsymbol{y} \lambda k_1 \lambda k_2 \exists x. \big((unicorn(x, k_2) \wedge find(x, \boldsymbol{y}^\circ, k_2)) \wedge$
$\big(\forall p. (p(k_1) \wedge (\exists z. abt(z, [\lambda k. find(x, \boldsymbol{y}^\circ, k)]) \wedge abt(z, p))) \twoheadrightarrow p(k_2)\big)\big)\big](\boldsymbol{x}, i, j)$
$= \lambda \boldsymbol{y} \lambda i \lambda j. \urcorner \big(\exists x. \big((unicorn(x, j) \wedge find(x, \boldsymbol{y}^\circ, j)) \wedge \big(\forall p. (p(i) \wedge (\exists z. abt(z, [\lambda k. find(x, \boldsymbol{y}^\circ, k)]) \wedge abt(z, p))) \twoheadrightarrow p(j)\big)\big)\big)$

6. $[_{\text{NP}}\text{John}] \rightsquigarrow \boldsymbol{john} = \lambda i \lambda j \big(\forall p. p(i) \wedge abt(p, john) \twoheadrightarrow p(j)\big)$

7. $[_{\text{S}}[_{\text{NP}}\text{John}][_{\text{VP}}[_{\text{ADV}}\text{does not}][_{\text{VP}}[_{\text{TV}}\text{find}][_{\text{NP}}[_{\text{DET}}\text{a}][_{\text{N}}\text{unicorn}]]]]] \rightsquigarrow \urcorner \big(\bigvee \boldsymbol{x}. \boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{find}(\boldsymbol{x}, \boldsymbol{john})\big)$
$= \lambda i \lambda j. \urcorner \big((\exists x. unicorn(x, j) \wedge find(x, john, j)) \wedge \big(\forall p. (p(i) \wedge (\exists z. abt(z, [\lambda k. find(y, john, k)]) \wedge abt(z, p))) \twoheadrightarrow p(j)\big)\big)$
$= \lambda i \lambda j. \big(\neg[\exists x. unicorn(x, j) \wedge find(x, john, j)] \wedge \big(\forall p. \big(p(i) \wedge \big(\exists y. abt\big(y, \big([\lambda k. \neg(\exists x. unicorn(x, k) \wedge$
$find(x, john, k))]\big) \wedge abt(y, p)\big)\big)\big) \twoheadrightarrow p(j)\big)\big)$

The definition of the STY_1^3 translation of the CP equative The problem is that Mary hates Bill (cf. (3.2.22)) is given below:

(8.2.22)

1. $[_{\mathrm{CP}}[_{\mathrm{C}}\mathsf{that}][_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{Mary}][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{hates}][_{\mathrm{NP}}\mathsf{Bill}]]]] \rightsquigarrow \boldsymbol{hate}(\boldsymbol{bill}, \boldsymbol{mary})$
   $= \lambda i \lambda j.\, hate\big(bill, mary, j\big) \wedge \big(\forall p.(p(i) \wedge (abt\,(bill, p) \vee abt\,(mary, p))) \twoheadrightarrow p(j)\big)$
2. $[_{\mathrm{TV}}\mathsf{is}] \rightsquigarrow \lambda \boldsymbol{Q} \lambda \boldsymbol{x}.\, \boldsymbol{Q}(\lambda \boldsymbol{y}.\, \boldsymbol{x} \doteq \boldsymbol{y})$
   $= \lambda \boldsymbol{Q} \lambda \boldsymbol{x}.\, \boldsymbol{Q}\,\big(\lambda \boldsymbol{y} \lambda i \lambda j.\, \boldsymbol{x}^\circ = \underline{\mathit{flex}\,(\boldsymbol{y}^\circ)} \wedge \big(\forall p.\big(p(i) \wedge \big(\exists x.\, abt\,\big(x, [\lambda k.\, \boldsymbol{x}^\circ = \underline{\mathit{flex}\,(\boldsymbol{y}^\circ)}]\big) \wedge abt(x, p)\big)\big) \twoheadrightarrow p(j)\big)\big)$
3. $[_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{is}][_{\mathrm{CP}}[_{\mathrm{C}}\mathsf{that}][_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{Mary}][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{hates}][_{\mathrm{NP}}\mathsf{Bill}]]]]]$
   $\rightsquigarrow \lambda \boldsymbol{x}.\, \boldsymbol{x} \doteq \boldsymbol{hate}(\boldsymbol{bill}, \boldsymbol{mary})$
   $= \lambda \boldsymbol{Q} \lambda \boldsymbol{x}.\, \boldsymbol{Q}\,\big(\lambda \boldsymbol{y} \lambda i \lambda j.\, \boldsymbol{x}^\circ = \underline{\mathit{flex}\,(\boldsymbol{y}^\circ)} \wedge \big(\forall p.\big(p(i) \wedge \big(\exists x.\, abt\,\big(x, [\lambda k.\, \boldsymbol{x}^\circ = \underline{\mathit{flex}\,(\boldsymbol{y}^\circ)}]\big) \wedge abt(x, p)\big)\big) \twoheadrightarrow p(j)\big)\big)$
   $\big[\mathit{lift}\,\big(\lambda k_1 \lambda k_2.\, hate\,\big(bill, mary, k_2\big) \wedge \big(\forall q.(q(k_1) \wedge (abt\,(bill, q) \vee abt\,(mary, q))) \twoheadrightarrow p(k_2)\big)\big)\big]$
   $= \lambda \boldsymbol{Q} \lambda \boldsymbol{x}.\, \boldsymbol{Q}\,\big(\lambda \boldsymbol{y} \lambda i \lambda j.\, \boldsymbol{x}^\circ = \underline{\mathit{flex}\,(\boldsymbol{y}^\circ)} \wedge \big(\forall p.\big(p(i) \wedge \big(\exists x.\, abt\,\big(x, [\lambda k.\, \boldsymbol{x}^\circ = \underline{\mathit{flex}\,(\boldsymbol{y}^\circ)}]\big) \wedge abt(x, p)\big)\big) \twoheadrightarrow p(j)\big)\big)$
   $\big[\lambda \boldsymbol{P}.\, \boldsymbol{P}\,\big(\lambda k_1 \lambda k_2.\, hate\,\big(bill, mary, k_2\big) \wedge \big(\forall q.(q(k_1) \wedge (abt\,(bill, q) \vee abt\,(mary, q))) \twoheadrightarrow p(k_2)\big)\big)\big]$
   $= \lambda \boldsymbol{x}.\big[\lambda \boldsymbol{P}.\, \boldsymbol{P}\,\big(\lambda k_1 \lambda k_2.\, hate\,\big(bill, mary, k_2\big) \wedge \big(\forall q.(q(k_1) \wedge (abt\,(bill, q) \vee abt\,(mary, q))) \twoheadrightarrow p(k_2)\big)\big)\big]$
   $\big(\lambda \boldsymbol{y} \lambda i \lambda j.\, \boldsymbol{x}^\circ = \underline{\mathit{flex}\,(\boldsymbol{y}^\circ)} \wedge \big(\forall p.\big(p(i) \wedge \big(\exists x.\, abt\,\big(x, [\lambda k.\, \boldsymbol{x}^\circ = \underline{\mathit{flex}\,(\boldsymbol{y}^\circ)}]\big) \wedge abt(x, p)\big)\big) \twoheadrightarrow p(j)\big)\big)$
   $= \lambda \boldsymbol{x}.\big[\big(\lambda \boldsymbol{y} \lambda i \lambda j.\, \boldsymbol{x}^\circ = \underline{\mathit{flex}\,(\boldsymbol{y}^\circ)} \wedge \big(\forall p.\big(p(i) \wedge \big(\exists x.\, abt\,\big(x, [\lambda k.\, \boldsymbol{x}^\circ = \underline{\mathit{flex}\,(\boldsymbol{y}^\circ)}]\big) \wedge abt(x, p)\big)\big) \twoheadrightarrow p(j)\big)\big)$
   $\big(\lambda k_1 \lambda k_2.\, hate\,\big(bill, mary, k_2\big) \wedge \big(\forall q.(q(k_1) \wedge (abt\,(bill, q) \vee abt\,(mary, q))) \twoheadrightarrow p(k_2)\big)\big)\big]$
   $= \lambda \boldsymbol{x} \lambda i \lambda j.\, \boldsymbol{x}^\circ = \underline{\mathit{flex}\,\big(\big[\lambda k_1 \lambda k_2.\, hate\,\big(bill, mary, k_2\big) \wedge \big(\forall q.(q(k_1) \wedge (abt\,(bill, q) \vee abt\,(mary, q))) \twoheadrightarrow p(k_2)\big)\big]^\circ\big)} \wedge$
   $\big(\forall p.\big(p(i) \wedge \big(\exists x.\, abt\,\big(x, \big[\lambda k.\, \boldsymbol{x}^\circ = \underline{\mathit{flex}\,\big(\big[\lambda k_1 \lambda k_2.\, hate\,\big(bill, mary, k_2\big) \wedge \big(\forall q.(q(k_1) \wedge (abt\,(bill, q) \vee abt\,(mary, q)))$
   $\twoheadrightarrow p(k_2)\big)\big]^\circ\big)\big]\big) \wedge abt(x, p)\big)\big) \twoheadrightarrow p(j)\big)$

4. $[_{\text{NP}}[_{\text{DET}}\textsf{the}][_{\text{N}}\textsf{problem}]] \rightsquigarrow \lambda \boldsymbol{P} \bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}.(\boldsymbol{problem}(\boldsymbol{y}) \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}) \wedge \boldsymbol{P}(\boldsymbol{x})$

   $= \lambda \boldsymbol{P} \lambda i \lambda j \exists \boldsymbol{x} \forall \boldsymbol{y}.\big(problem(\boldsymbol{y}^\circ, j) \wedge \big(\forall p.(p(i) \wedge (\exists y.\, abt(y, [\lambda k.\, problem(\boldsymbol{y}^\circ, j)]) \wedge abt(y, p))) \twoheadrightarrow p(i)\big)\big) \leftrightarrow$
   $\boldsymbol{x}(i,j) = \boldsymbol{y}(i,j)) \wedge \boldsymbol{P}(\boldsymbol{x}, i, j)$

5. $[_{\text{S}}[_{\text{NP}}[_{\text{DET}}\textsf{the}][_{\text{N}}\textsf{problem}]][_{\text{VP}}[_{\text{TV}}\textsf{is}][_{\text{CP}}[_{\text{C}}\textsf{that}][_{\text{S}}[_{\text{NP}}\textsf{Mary}][_{\text{VP}}[_{\text{TV}}\textsf{hates}][_{\text{NP}}\textsf{Bill}]]]]]]$

   $\rightsquigarrow \bigvee \boldsymbol{x} \bigwedge \boldsymbol{y}.(\boldsymbol{problem}(\boldsymbol{y}) \dot{\leftrightarrow} \boldsymbol{x} \doteq \boldsymbol{y}) \wedge \boldsymbol{x} \doteq \boldsymbol{hate}(\boldsymbol{bill}, \boldsymbol{mary})$

   $= \lambda \boldsymbol{P} \lambda i \lambda j \exists \boldsymbol{x} \forall \boldsymbol{y}.\big(problem(\boldsymbol{y}^\circ, j) \wedge \big(\forall p.(p(i) \wedge (\exists y.\, abt(y, [\lambda k.\, problem(\boldsymbol{y}^\circ, j)]) \wedge abt(y, p))) \twoheadrightarrow p(i)\big)\big) \leftrightarrow \boldsymbol{x}(i,j) =$
   $\boldsymbol{y}(i,j)) \wedge \boldsymbol{P}(\boldsymbol{x}, i, j)\, \big[\lambda \boldsymbol{z}.\big(\lambda k_3 \lambda k_4.\, \boldsymbol{z}^\circ = \underline{\mathit{flex}\big(\big[\lambda k_1 \lambda k_2.\, hate\,\big(bill, mary, k_2\big) \wedge \big(\forall q.(q(k_1) \wedge (abt(bill, q) \vee}$
   $\underline{abt(mary, q))) \twoheadrightarrow p(k_2)\big)\big]^\circ\big)} \wedge \big(\forall r.\big(r(k_3) \wedge \big(\exists z.\, abt\,\big(z, \big[\lambda k.\, \boldsymbol{z}^\circ = \underline{\mathit{flex}\big(\big[\lambda k_1 \lambda k_2.\, hate\,\big(bill, mary, k_2\big) \wedge}$
   $\underline{\big(\forall q.(q(k_1) \wedge (abt(bill, q) \vee abt(mary, q))) \twoheadrightarrow p(k_2)\big)\big]^\circ\big)}\big]\big) \wedge abt(x, p)\big)\big) \twoheadrightarrow p(k_3)\big)\big)\big]$

   $= \lambda i \lambda j \exists \boldsymbol{x} \forall \boldsymbol{y}.\big(problem(\boldsymbol{y}^\circ, j) \wedge \big(\forall p.(p(i) \wedge (\exists y.\, abt(y, [\lambda k.\, problem(\boldsymbol{y}^\circ, j)]) \wedge abt(y, p))) \twoheadrightarrow p(i)\big)\big) \leftrightarrow \boldsymbol{x}(i,j) = \boldsymbol{y}(i,j)) \wedge$
   $\big[\lambda \boldsymbol{z}.\big(\lambda k_3 \lambda k_4.\, \boldsymbol{z}^\circ = \underline{\mathit{flex}\big(\big[\lambda k_1 \lambda k_2.\, hate\,\big(bill, mary, k_2\big) \wedge \big(\forall q.(q(k_1) \wedge (abt(bill, q) \vee abt(mary, q))) \twoheadrightarrow p(k_2)\big)\big]^\circ\big)} \wedge$
   $\big(\forall r.\big(r(k_3) \wedge \big(\exists z.\, abt\,\big(z, \big[\lambda k.\, \boldsymbol{z}^\circ = \underline{\mathit{flex}\big(\big[\lambda k_1 \lambda k_2.\, hate\,\big(bill, mary, k_2\big) \wedge \big(\forall q.(q(k_1) \wedge (abt(bill, q) \vee abt(mary, q)))}$
   $\underline{\twoheadrightarrow p(k_2)\big)\big]^\circ\big)}\big]\big) \wedge abt(x, p)\big)\big) \twoheadrightarrow p(k_3)\big)\big)\big](\boldsymbol{x}, i, j)$

   $= \lambda i \lambda j \exists \boldsymbol{x} \forall \boldsymbol{y}.\big(problem(\boldsymbol{y}^\circ, j) \wedge \big(\forall p.(p(i) \wedge (\exists y.\, abt(y, [\lambda k.\, problem(\boldsymbol{y}^\circ, j)]) \wedge abt(y, p))) \twoheadrightarrow p(i)\big) \leftrightarrow \boldsymbol{x}(i,j) = \boldsymbol{y}(i,j)\big) \wedge$
   $\big[\boldsymbol{x}^\circ = \underline{\mathit{flex}\big(\big[\lambda k_1 \lambda k_2.\, hate\,\big(bill, mary, k_2\big) \wedge \big(\forall q.(q(k_1) \wedge (abt(bill, q) \vee abt(mary, q))) \twoheadrightarrow p(k_2)\big)\big]^\circ\big)} \wedge \big(\forall r.\big(r(i) \wedge$
   $\big(\exists z.\, abt\,\big(z, \big[\lambda k.\, \boldsymbol{x}^\circ = \underline{\mathit{flex}\big(\big[\lambda k_1 \lambda k_2.\, hate\,\big(bill, mary, k_2\big) \wedge \big(\forall q.(q(k_1) \wedge (abt(bill, q) \vee abt(mary, q))) \twoheadrightarrow}$
   $\underline{p(k_2)\big)\big]^\circ\big)}\big]\big) \wedge abt(x, p)\big)\big) \twoheadrightarrow p(j)\big)\big]$

   $= \lambda i \lambda j \exists x \forall y.\big(problem(y, j) \wedge \big(\forall p.(p(i) \wedge (\exists z.\, (abt(z, [\lambda k.\, problem(y, j)]) \wedge abt(y, p)))) \twoheadrightarrow p(i)\big) \leftrightarrow x = y\big) \wedge$
   $\big(x = \big[\iota x_1.(\lambda k_3.\, E(x_1, k_3)) = [\lambda k_2.\, hate(bill, mary, k_2)]\big] \wedge \big(\forall p.\big(p(i) \wedge \big(\exists z.\big(abt\,\big(z, \big[\lambda k.$
   $x = \big[\iota x_1.(\lambda k_3.\, E(x_1, k_3)) = [\lambda k_2.\, hate(bill, mary, k_2)]\big]\big]\big) \wedge abt(z, p)\big)\big)\big) \twoheadrightarrow p(j)\big)\big)$

The STY_1^3 translation of the wide-scope reading of the sentence John seeks a unicorn (cf. (3.2.14)) is defined as follows:

(8.2.11)

1. $[_{\mathrm{TV}}\mathsf{seeks}] \rightsquigarrow \boldsymbol{seek} = \lambda\boldsymbol{Q}\lambda\boldsymbol{x}\lambda i\lambda j.seek(\boldsymbol{Q}^\circ, \boldsymbol{x}^\circ, j) \wedge \big(\forall p.(p(i) \wedge (\exists z.abt(z, [\lambda k.seek(\boldsymbol{Q}^\circ, \boldsymbol{x}^\circ, k)]) \wedge abt(z, p))) \twoheadrightarrow p(j)\big)$
2. $t_0 \rightsquigarrow \boldsymbol{x}_0$
3. $[_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{seeks}]\ t_0] \rightsquigarrow \lambda\boldsymbol{y}.\boldsymbol{seek}\big([\lambda\boldsymbol{P}.\boldsymbol{P}(\boldsymbol{x}_0)], \boldsymbol{y}\big)$
   $= \lambda\boldsymbol{Q}\lambda\boldsymbol{x}\lambda i\lambda j.seek(\boldsymbol{Q}^\circ, \boldsymbol{x}^\circ, j) \wedge \big(\forall p.(p(i) \wedge (\exists z.abt(z, [\lambda k.seek(\boldsymbol{Q}^\circ, \boldsymbol{x}^\circ, k)]) \wedge abt(z, p))) \twoheadrightarrow p(j)\big)\ [\lambda\boldsymbol{P}.\boldsymbol{P}(\boldsymbol{x}_0)]$
   $= \lambda\boldsymbol{x}\lambda i\lambda j.seek([\lambda\boldsymbol{P}.\boldsymbol{P}(\boldsymbol{x}_0)]^\circ, \boldsymbol{x}^\circ, j) \wedge \big(\forall p.(p(i) \wedge (\exists z.abt(z, [\lambda k.seek([\lambda\boldsymbol{P}.\boldsymbol{P}(\boldsymbol{x}_0)]^\circ, \boldsymbol{x}^\circ, k)]) \wedge abt(z, p))) \twoheadrightarrow p(j)\big)$
4. $[_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{John}][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{seeks}]\ t_0]] \rightsquigarrow \boldsymbol{seek}\big([\lambda\boldsymbol{P}.\boldsymbol{P}(\boldsymbol{x}_0)], \boldsymbol{john}\big)$
   $= \lambda i\lambda j.seek([\lambda\boldsymbol{P}.\boldsymbol{P}(\boldsymbol{x}_0)]^\circ, john, j) \wedge \big(\forall p.(p(i) \wedge (\exists z.abt(z, [\lambda k.seek([\lambda\boldsymbol{P}.\boldsymbol{P}(\boldsymbol{x}_0)]^\circ, john, k)]) \wedge abt(z, p))) \twoheadrightarrow p(j)\big)$
5. $[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{unicorn}]]^0 \rightsquigarrow \lambda\boldsymbol{P}\bigvee\boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{P}(\boldsymbol{x})$
   $= \lambda\boldsymbol{P}\lambda i\lambda j\exists\boldsymbol{x}.\big(unicorn(\boldsymbol{x}^\circ, j) \wedge \big(\forall p.(p(i) \wedge (\exists y.abt(y, [\lambda k.unicorn(\boldsymbol{x}^\circ, j)]) \wedge abt(y, p))) \twoheadrightarrow p(i)\big)\big) \wedge \boldsymbol{P}(\boldsymbol{x}, i, j)$
6. $[_{\mathrm{S}}[_{\mathrm{NP}}[_{\mathrm{DET}}\mathsf{a}][_{\mathrm{N}}\mathsf{unicorn}]]^0\,[_{\mathrm{S}}[_{\mathrm{NP}}\mathsf{John}][_{\mathrm{VP}}[_{\mathrm{TV}}\mathsf{seeks}]\ t_0]]] \rightsquigarrow \bigvee\boldsymbol{x}.\boldsymbol{unicorn}(\boldsymbol{x}) \wedge \boldsymbol{seek}\big([\lambda\boldsymbol{P}.\boldsymbol{P}(\boldsymbol{x})], \boldsymbol{john}\big)$
   $= \lambda\boldsymbol{P}\lambda j\exists\boldsymbol{x}.\big(unicorn(\boldsymbol{x}^\circ, j) \wedge \big(\forall p.(p(i) \wedge (\exists y.abt(y, [\lambda k.unicorn(\boldsymbol{x}^\circ, j)]) \wedge abt(y, p))) \twoheadrightarrow p(i)\big)\big) \wedge \boldsymbol{P}(\boldsymbol{x}, i, j)\big[\lambda\boldsymbol{x}_0\lambda k_1$
   $\lambda k_2.seek([\lambda\boldsymbol{P}.\boldsymbol{P}(\boldsymbol{x}_0)]^\circ, john, k_2) \wedge \big(\forall q.(q(k_1) \wedge (\exists z.abt(z, [\lambda k.seek([\lambda\boldsymbol{P}.\boldsymbol{P}(\boldsymbol{x}_0)]^\circ, john, k)]) \wedge abt(z, q))) \twoheadrightarrow q(k_2)\big)\big]$
   $= \lambda j\exists\boldsymbol{x}.\big(unicorn(\boldsymbol{x}^\circ, j) \wedge \big(\forall p.(p(i) \wedge (\exists y.abt(y, [\lambda k.unicorn(\boldsymbol{x}^\circ, j)]) \wedge abt(y, p))) \twoheadrightarrow p(i)\big)\big) \wedge \big[\lambda\boldsymbol{x}_0\lambda k_1\lambda k_2.seek([\lambda\boldsymbol{P}.$
   $\boldsymbol{P}(\boldsymbol{x}_0)]^\circ, john, k_2) \wedge \big(\forall q.(q(k_1) \wedge (\exists z.abt(z, [\lambda k.seek([\lambda\boldsymbol{P}.\boldsymbol{P}(\boldsymbol{x}_0)]^\circ, john, k)]) \wedge abt(z, q))) \twoheadrightarrow q(k_2)\big)\big](\boldsymbol{x}, i, j)$
   $= \lambda i\lambda j\exists\boldsymbol{x}.\big(unicorn(\boldsymbol{x}^\circ, j) \wedge \big(\forall p.(p(i) \wedge (\exists y.abt(y, [\lambda k.unicorn(\boldsymbol{x}^\circ, j)]) \wedge abt(y, p))) \twoheadrightarrow p(i)\big)\big) \wedge \big(seek([\lambda\boldsymbol{P}.\boldsymbol{P}(\boldsymbol{x})]^\circ,$
   $john, j) \wedge \big(\forall q.(q(i) \wedge (\exists z.abt(z, [\lambda k.seek([\lambda\boldsymbol{P}.\boldsymbol{P}(\boldsymbol{x})]^\circ, john, j)]) \wedge abt(z, q)))) \twoheadrightarrow q(j)\big)\big)$
   $= \exists x.\big(unicorn(x, j) \wedge seek([\lambda P\lambda k.P(x, k)], john, j)\big) \wedge \big(\forall p.(p(i) \wedge (abt(x, p) \vee abt(john, p))) \twoheadrightarrow p(j)\big)$

Bibliography

Weber, Robert and Roger Gryson (eds.) *Biblia Sacra*: *Iuxta vulgatam versionem*, Deutsche Bibelgesellschaft, Stuttgart.

Bach, Emmon. 1986. *Natural language metaphysics*, Logic, Methodology and Philosophy of Science (Ruth Barcan Marcus, George J.W. Dorn, and Paul Weingartner, eds.), Vol. VII, Elsevier Science Publishers, Amsterdam, 1986.

Barendregt, Henk. 1984. *The Lambda Calculus*: *Its syntax and semantics*, Vol. 2, North-Holland, Amsterdam.

Barendregt, Henk and Erik Barendsen. 2000. *Introduction to the Lambda Calculus*: *Revised edition*, Technical report, University of Nijmegen.

Barendregt, Henk, Wil Dekkers, and Richard Statman. 2010. *Lambda Calculus with Types*, Perspectives in Logic, Cambridge University Press and ASL, Cambridge.

Barker, Chris. 2014. *Lambda Tutorial*, `https://files.nyu.edu/cb125/public/Lambda/`.

Barwise, Jon. 1981. *Scenes and other situations*, The Journal of Philosophy **78/7**, 369–397.

Barwise, Jon and John Perry. 1983. *Situations and Attitudes*, MIT Press, Cambridge, Mass.

———. 1985. *Shifting situations and shaken attitudes*, Linguistics and Philosophy **8**, 105–161.

Barwise, Jon. 1997. *Information and impossibilities*, Notre Dame Journal of Formal Logic **38/4**, 488–515.

Bayer, Samuel. 1996. *The coordination of unlike categories*, Language **72/3**, 579–616.

Bekki, Daisuke. 2010. *Nihongobunpou-no Keishikiriron*: *Katsuyoutaikei, tougokouzou, imigousei* [*Formal Theory of Japanese Grammar: The conjugation system, categorial syntax, and dynamic semantics*], Kuroshio, Tokyo.

———. 2012. *Dependent type semantics*: *The framework*, Manuscript, Ochanomizu University, Tokyo.

Belnap, Nuel D. 1977. *A useful four-valued logic*, Modern Uses of Multiple-Valued Logics (J. Michael Dunn and George Epstein, eds.), Reidel, Dordrecht, 1977.

van Benthem, Johan. 1989. *Semantic parallels in natural language and computation*, Logic Colloquium: Granada 1987 (H.-D. Ebbinghaus et al., ed.), Elsevier, Amsterdam, 1989.

van Benthem, Johan and Kees Doets. 2001. *Higher-order logic*, Handbook of Philosophical Logic (Dov M. Gabbay and Franz Guenthner, eds.), Vol. 1, Kluwer, Dordrecht, 2001.

Blamey, Stephen. 1986. *Partial logic*, Handbook of Philosophical Logic (Dov M. Gabbay and Franz Guenthner, eds.), Vol. 5, Kluwer Academic Publishers, 2002.

Ben-Avi, Gilad and Yoad Winter. 2007. *The semantics of intensionalization*, Workshop on New Directions in Type-Theoretic Grammars (Reinhard Muskens, ed.), 2007.

Borg, Emma. 2004. *Minimal Semantics*, Oxford University Press, Oxford and New York.

———. 2005. *Saying what you mean*: *Unarticulated constituents and communication*, Ellipsis and Nonsentential Speech (Reinaldo Elugardo and Robert J. Stainton, eds.), Studies in Linguistics and Philosophy, Springer, Dordrecht, 2005.

Cappelen, Herman and Ernie Lepore. 2005. *Insensitive Semantics*: *A defense of semantic minimalism and speech act pluralism*, Blackwell, Oxford.
Carnap, Rudolf. 1988. *Meaning and Necessity*: *A study in semantics and modal logic*, University of Chicago Press, Chicago and London.
Carroll, Lewis. 2000. *Through the Looking-Glass and What Alice Found There*, The Annotated Alice: The definitive edition (Martin Gardner, ed.), W. W. Norton, New York, 2000.
Carstairs-McCarthy, Andrew. 1999. *The Origins of Complex Language*: *An inquiry into the evolutionary beginnings of sentences, syllables, and truth*, Oxford University Press, Oxford.
———. 2005. *The link between sentences and 'assertion'*: *An evolutionary accident?*, Ellipsis and Nonsentential Speech (Reinaldo Elugardo and Robert J. Stainton, eds.), Studies in Linguistics and Philosophy, Springer, Dordrecht, 2005.
Cheney, Dorothy L. and Robert M. Seyfarth. 1990. *How Monkeys See the World*: *Inside the mind of another species*, University of Chicago Press, Chicago.
Chierchia, Gennaro and Raymond Turner. 1988. *Semantics and property theory*, Linguistics and Philosophy **11**, 261–302.
Chierchia, Gennaro. 1998. *Reference to kinds across languages*, Natural Language Semantics **6**, 339–405.
Chierchia, Gennaro, Barbara Partee, and Raymond Turner (eds.) 1988a. *Properties, Types, and Meaning*: *Volume I: Foundational issues*, Studies in Linguistics and Philosophy, vol. 38, Kluwer Academic Publishers, Dordrecht, Boston, and London.
——— (ed.) 1988b. *Properties, Types, and Meaning*: *Volume II: Semantic issues*, Studies in Linguistics and Philosophy, vol. 39, Kluwer Academic Publishers, Dordrecht, Boston, and London.
Chierchia, Gennaro and Sally McConnell-Ginet. 2000. *Meaning and Grammar*: *An introduction to semantics*, MIT Press, Cambridge, Mass.
Chomsky, Noam. 1981. *Lectures on Government and Binding*, Foris, Dordrecht.
Church, Alonzo. 1932. *A set of postulates for the foundation of logic I*, Annals of Mathematics **33**, 346–366.
———. 1940. *A formulation of the Simple Theory of Types*, Journal of Symbolic Logic **5/2**, 56–68.
———. 1985. *The Calculi of Lambda Conversion*, Princeton University Press, Princeton, NJ.
Cresswell, Maxwell J. 1976. *The semantics of degree*, Montague Grammar (Barbara Partee, ed.), Academic Press, New York, 1976.
Culicover, Peter W. and Ray Jackendoff. 2005. *Simpler Syntax*, Oxford University Press, Oxford and New York.
Curry, Haskell B. 1934. *Functionality in combinatory logic*, Proceedings of the National Academy of Science of the USA **20**, 584–590.
Dalrymple, Mary. 2005. *Against reconstruction in ellipsis*, Ellipsis and Nonsentential Speech (Reinaldo Elugardo and Robert J. Stainton, eds.), Studies in Linguistics and Philosophy, Springer, Dordrecht, 2005.
Davey, B. A. and H. A. Priestley. 2002. *Introduction to Lattices and Order*: *Second Edition*, Cambridge University Press, Cambridge and New York.
Davidson, Donald. 1967. *The logical form of action sentences*, Essays on Actions and Events: Philosophical essays of Donald Davidson, Clarendon Press, Oxford and New York, 2001.
Dowty, David R. 1979. *Word Meaning and Montague Grammar*: *The semantics of verbs and times in generative semantics and in Montague's PTQ*, Synthese Language Library, vol. 7, D. Reidel Publishing Company, Dordrecht.
Dowty, David R., Robert E. Wall, and Stanley Peters. 1981. *Introduction to Montague Semantics*, Studies in Linguistics and Philosophy, vol. 11, Kluwer Academic Publishers, Dordrecht.

Dunn, J. Michael. 1976. *Intuitive semantics for first-degree entailments and 'coupled trees'*, Philosophical Studies **29**, 149–168.
van Eijck, Jan and Christina Unger. 2010. *Computational Semantics with Functional Programming*, Cambridge University Press, Cambridge and New York.
Elugardo, Reinaldo and Robert J. Stainton (eds.) 2005. *Ellipsis and Nonsentential Speech*, Studies in Linguistics and Philosophy, vol. 81, Springer, Dordrecht.
Fine, Kit. 1982. *First-order modal theories III*: *Facts*, Synthese **53**, 43–122.
Fox, Chris, Shalom Lappin, and Carl Pollard. 2002. *A higher-order fine-grained logic for intensional semantics*, Proceedings of the 7. International Symposium on Logic and Language, 2002.
Fox, Chris and Shalom Lappin. 2004. *An expressive first-order logic with flexible typing for natural language semantics*, Logic Journal of the IGPL **12/2**, 135–168.
_____. 2005. *Foundations of Intensional Semantics*, Blackwell, Malden, Mass.
Frege, Gottlob. 1891. *Funktion und Begriff*, Gottlob Frege: Kleine Schriften, Edition Classic Verlag Dr. Müller, Saarbrücken, 2006, pp. 125–142.
_____. 1892. *Ueber Begriff und Gegenstand*: *Eine kritische Auseinandersetzung mit Kerry*, Nachgelassene Schriften (Hans Hermes, Friedrich Kambartel, and Friedrich Kaulbach, eds.), Felix Meiner, Hamburg, 1969, pp. 96–127.
Friedman, Harvey. 1975. *Some systems of second order arithmetic and their use*, Proceedings of the International Congress of Mathematicians, Vol. 1, Canadian Mathematical Congress, Montreal, 1974.
Gallin, Daniel. 1975. *Intensional and Higher-Order Modal Logic with Applications to Montague Semantics*, North Holland, Amsterdam.
Gamut, L.T.F. 1991. *Intensional Logic and Logical Grammar*, Logic, Language, and Meaning, vol. 2, University of Chicago Press, Chicago and London.
Gazdar, Gerald. 1980. *A cross-categorial semantics for coordination*, Linguistics and Philosophy **3**, 407–409.
Girard, Jean-Yves. 1972. *Interprétation Fonctionnelle et Élimination des Coupures de l'Arithmétique d'Ordre Supérieur*, Doctoral dissertation, Université Paris VII, Paris.
Goodman, Nelson. 1972. *About*, Problems and Projects, Bobbs-Merrill, Indianapolis, 1972.
Groenendijk, Jeroen and Martin Stokhof. 1984. *Studies on the Semantics of Questions and the Pragmatics of Answers*, Doctoral dissertation, University of Amsterdam, Amsterdam.
_____. 1991. *Two theories of dynamic semantics*, Lecture Notes in Computer Science **478**, 55-64.
Groenendijk, Jeroen and Floris Roelofsen. 2009. *Inquisitive semantics and pragmatics*, Meaning, content, and argument: Proceedings of the ILCLI International Workshop on Semantics, Pragmatics and Rhetoric, Universidad del País Vasco, 2009.
de Groote, Philippe and Makoto Kanazawa. 2013. *A note on intensionalization*, Journal of Logic, Language and Information **22/2**, 173–194.
Harley, Heidi. forthcoming. *Semantics in distributed morphology*, Semantics: An International Handbook of Natural Language Meaning (Claudia Maienborn, Klaus von Heusinger, and Paul Portner, eds.), Vol. 3, Mouton de Gruyter, Berlin, forthcoming.
Harley, Heidi and Rolf Noyer. 2003. *Distributed morphology*, The Second Glot International State-of-the-Article Book (Lisa Cheng and Rint Sybesma, eds.), Mouton de Gruyter, Berlin, 2003.
Heim, Irene and Angelika Kratzer. 1998. *Semantics in Generative Grammar*, Blackwell Textbooks in Linguistics, vol. 13, Blackwell, Malden, Mass. and Oxford.
Hendriks, Herman. 1990. *Flexible Montague Grammar*, ITLI Prepublication Series for Logic, Semantics and Philosophy of Language **08**.
Henkin, Leon. 1950. *Completeness in the theory of types*, Journal of Symbolic Logic **15**, 81–91.

———. 1963. *A theory of propositional types*, Fundamenta Mathematicae **52**, 323–344.
Heycock, Caroline and Anthony Kroch. 1999. *Pseudocleft connectedness*: *Implications for the LF interface level*, Linguistic Inquiry **30**, 365–397.
Hill, Patricia M. and Rodney W. Topor. 1992. *A semantics for typed logic programs*, Types in Logic Programming (Frank Pfenning, ed.), Logic Programming, MIT Press, Cambridge, 1992.
Hindley, J. Roger and Jonathan P. Seldin. 2008. *Lambda-Calculus and Combinators*: *An introduction*, Cambridge University Press, Cambridge.
Hintikka, Jaakko. 1975. *Impossible possible worlds vindicated*, Journal of Philosophical Logic **4**, 475–484.
Hodges, Wilfrid. 2001. *Formal features of compositionality*, Journal of Logic, Language and Information **10**, 7–28.
Janssen, Theo M.V. 1986. *Foundations and Applications of Montague Grammar*, CWI Tracts, vol. 19, 28, CWI, Amsterdam.
———. 1997. *Compositionality*, Handbook of Logic and Language (Johan van Benthem and Alice G.B. ter Meulen, eds.), Elsevier Science Publishers, Amsterdam, 1997.
———. 2012. *Montague Semantics*, The Stanford Encyclopedia of Philosophy, Winter 2012 edition (Edward N. Zalta, ed.), 2012.
Kaplan, David. 1989. *Demonstratives*: *An essay on the semantics, logic, metaphysics, and epistemology of demonstratives and other indexicals*, Themes from Kaplan (Joseph Almog, John Perry, and Howard Wettstein, eds.), Oxford University Press, Oxford and New York, 1989.
Kim, Jong-Bok. 2008. *Grammatical interfaces in English object extraposition*, Linguistic Research **25/3**, 117–131.
Kim, Jong-Bok and Ivan A. Sag. 2005. *It-extraposition in English*: *A constraint-based approach*, HPSG'2005. The 12th International Conference on Head-Driven Phrase Structure Grammar: Conference Notes (António Branco, Francisco Costa, and Manfred Sailer, eds.), Technical Report, Universidade de Lisboa, Lisbon, 2005.
Kleene, Stephen Cole. 1938. *On notation for ordinal numbers*, Journal of Symbolic Logic **3/4**, 150-155.
Klein, Ewan and Ivan Sag. 1985. *Type-driven translation*, Linguistics and Philosophy **8**, 163–201.
Koster, Jan. 2000. *Extraposition as Parallel Construal*, Manuscript, University of Groningen.
Kraak, A. and Wim Klooster. 1968. *Syntaxis*, Stam-Kemperman, Culemborg.
Kratzer, Angelika. 1989. *An investigation into the lumps of thought*, Linguistics and Philosophy **12**, 607–653.
———. 1990. *How specific is a fact?*, Proceedings of the Conference on Theories of Partial Information, Center for Cognitive Science, University of Texas, Austin, 1990.
———. 2002. *Facts: particulars or information units?*, Linguistics and Philosophy **25/5-6**, 655–670.
———. 2011. *Situations in natural language semantics*, The Stanford Encyclopedia of Philosophy, Fall 2011 edition (Edward N. Zalta, ed.), 2011.
Kripke, Saul A. 1959. *A completeness theorem in modal logic*, Journal of Symbolic Logic **24/1**, 1–14.
———. 1963. *Semantical considerations on modal logic*, Acta Philosophica Fennica **16**, 83–94.
———. 1980. *Naming and Necessity*, Harvard University Press, Cambridge, Mass.
Lakoff, George. 1970. *Linguistics and natural logic*, Synthese **22**, 151–271.
Lambek, Joachim. 1958. *The mathematics of sentence structure*, American Mathematical Monthly **65**, 154–170.
Lambek, Joachim and P. Scott. 1986. *Introduction to Higher-Order Categorical Logic*, Cambridge Studies in Advanced Mathematics, vol. 7, Cambridge University Press, Cambridge.

Lambert, Karel. 1960. *The definition of E! in free logic*, Abstracts: The International Congress for Logic, Methodology and Philosophy of Science, Stanford University Press, Stanford, 1960.
Landman, Fred. 1985. *Data semantics: An epistemic theory of partial objects*, Towards a Theory of Information: The status of partial objects in semantics (Jeroen Groenendijk, Dick de Jongh, and Martin Stokhof, eds.), Groningen-Amsterdam Studies in Semantics, vol. 8, Foris Publications, Dordrecht, 1986.
_____. 1984. *Pegs and alecs*, Towards a Theory of Information: The status of partial objects in semantics (Jeroen Groenendijk, Dick de Jongh, and Martin Stokhof, eds.), Groningen-Amsterdam Studies in Semantics, vol. 8, Foris Publications, Dordrecht, 1986.
_____. 1986. *Towards a Theory of Information: The status of partial objects in semantics* (Jeroen Groenendijk, Dick de Jongh, and Martin Stokhof, eds.), Groningen-Amsterdam Studies in Semantics, vol. 8, Foris Publications, Dordrecht.
_____. 1991. *Structures for Semantics* (Gennaro Chierchia, Pauline Jacobson, and Francis J. Pelletier, eds.), Studies in Linguistics and Philosophy, vol. 45, Kluwer Academic Publishers, Dordrecht, Boston, and London.
Lewis, David. 1986. *On the Plurality of Worlds*, Basil Blackwell, Oxford.
Liefke, Kristina and Stephan Hartmann. 2011. *Integrative reduction, confirmation, and the syntax-semantics map*, Manuscript, Tilburg Center for Logic and Philosophy of Science, *PhilSci Archive*, `http://philsci-archive.pitt.edu/8760/`.
Liefke, Kristina. 2013. *A single-type logic for natural language*, Journal of Logic and Computation: Special issue for CiE 2010 (Alessandra Carbone, Fernando Ferreira, Benedikt Löwe, and Elvira Mayordomo, eds.), online first, `http://logcom.oxfordjournals.org/content/early/2013/01/23/logcom.exs074.short`, 2013.
Link, Godehard. 1983. *The logical analysis of plurals and mass terms: A lattice-theoretical approach*, Formal Semantics: The Essential Readings (Paul Portner and Barbara Partee, eds.), Blackwell, Oxford and Malden, Mass., 2002.
Linsky, Bernard and Edward N. Zalta. 1994. *In defense of the simplest quantified modal logic*, Philosophical Perspectives **8**, 431–458.
Löbner, Sebastian. 1976. *Einführung in die Montague Grammatik*, Scriptor Verlag, Kronberg.
Lycan, William G. 2008. *Philosophy of Language: A contemporary introduction*, Routledge Contemporary Introductions to Philosophy, Routledge, New York.
Martinich, A. P. and David Sosa (eds.) 2012. *The Philosophy of Language*, Oxford University Press, Oxford and New York.
Martin-Löf, Per. 1973. *An intuitionistic theory of types*, Logic Colloquium '73 (H.E. Rose and J. Shepherdson, eds.), Studies in Logic and the Foundations of Mathematics, vol. 80, North-Holland, Amsterdam, 1975.
May, Robert. 1977. *The Grammar of Quantification*, Dissertation, MIT, Cambridge, Mass.
_____. 1985. *Logical Form*, MIT Press, Cambridge, Mass.
Merchant, Jason. 2008. *Three types of ellipsis*, Context-Dependence, Perspective, and Relativity (F. Recanati, I. Stojanovic, and N. Villanueva, eds.), Mouton de Gruyter, Berlin, 2010.
Montague, Richard. 1970a. *English as a formal language*, Formal Philosophy: Selected papers of Richard Montague (Richmond H. Thomason, ed.), Yale University Press, New Haven and London, 1976.
_____. 1970b. *Universal grammar*, Formal Philosophy: Selected papers of Richard Montague (Richmond H. Thomason, ed.), Yale University Press, New Haven and London, 1976.
_____. 1973. *The proper treatment of quantification in ordinary English*, Formal Philosophy: Selected papers of Richard Montague (Richmond H. Thomason, ed.), Yale University Press, New Haven and London, 1976.

Muskens, Reinhard. 1989. *A relational formulation of the theory of types*, Linguistics and Philosophy **12**, 325–346.
_____. 1991. *Anaphora and the logic of change*, Logic in AI **478**, 412–427.
_____. 1995. *Meaning and Partiality*, CSLI Lecture Notes, FoLLI, Stanford.
_____. 1996. *Combining Montague semantics and discourse representation*, Linguistics and Philosophy **19**, 143–186.
_____. 1999. *On partial and paraconsistent logics*, Notre Dame Journal of Formal Logic **40/3**, 352–374.
_____. 2005. *Sense and the computation of reference*, Linguistics and Philosophy **28**, 473-504.
_____. 2007. *Intensional models for the theory of types*, Journal of Symbolic Logic **72/1**, 98–118.
_____. 2011. *Type-logical semantics*, Routledge Encyclopedia of Philosophy Online (Edward Craig, ed.), Routledge, New York, 2011.
Nicod, Jean G. 1917. *A reduction in the number of primitive propositions of logic*, Proceedings of the Cambridge Philosophical Society **19**, 32–41.
Niles, Ian and Adam Pease. 2001. *Towards a standard upper ontology*, Formal Ontology in Information Systems: Proceedings of the first international conference, ACM, New York, 2001.
Orey, Steven. 1959. *Model theory for the higher order predicate calculus*, Transactions of the American Mathematical Society **92/1**, 72–84.
Parsons, Terence. 1990. *Events in the Semantics of English*: *A study in subatomic semantics*, Current Studies in Linguistics, vol. 19, MIT Press, Cambridge, Mass.
Partee, Barbara and Mats Rooth. 1983. *Generalized conjunction and type ambiguity*, Meaning, Use and Interpretation of Language (Rainer Bauerle, Christoph Schwarze, and Arnim von Stechow, eds.), Foundations of Communication, Walter De Gruyter, Berlin, 1983.
Partee, Barbara. 1984. *Compositionality*, Varieties of Formal Semantics: Proceedings of the 4th Amsterdam Colloquium (Fred Landman and Frank Veltman, eds.), Groningen-Amsterdam Studies in Semantics, vol. 3, 1984.
_____. 1987. *Noun phrase interpretation and type-shifting principles*, Studies in Discourse Representation Theory and the Theory of Generalized Quantifiers (Jeroen Groenendijk, Dick de Jong, and Martin Stokhof, eds.), Foris Publications, Dordrecht, 1987.
_____. 1996. *The development of formal semantics*, The Handbook of Contemporary Semantic Theory (Shalom Lappin, ed.), Blackwell, Oxford, 1996.
_____. 1997. *Montague grammar*, Handbook of Logic and Language (Johan van Benthem and Alice G. B. ter Meulen, eds.), Elsevier Science Publishers, Amsterdam, 1997.
_____. 2001. *Montague grammar*, International Encyclopedia of the Social and Behavioral Sciences (Neil J. Smelser and Paul B. Baltes, eds.), Pergamon and Elsevier, Oxford, 2001.
_____. 2006. *Do we need two basic types?*, Snippets: Special issue in honor of Manfred Krifka (Sigrid Beck and Hans-Martin Gärtner, eds.), Vol. 20, Berlin, 2009.
Beck, Sigrid and Hans-Martin Gärtner (eds.) 2009. *Snippets*: *Special Issue in Honor of Manfred Krifka*, Vol. 20, Berlin.
Perry, John. 1986. *Possible worlds and subject matter*: *Discussion of Barbara H. Partee's paper "Possible worlds in model-theoretic semantics: A linguistic perspective"*, Possible Worlds in Humanities, Arts and Sciences: Proceedings of Nobel Symposium 65 (Alléen Sture, ed.), Research in Text Theory, vol. 14, Walter de Gruyter, Berlin and New York, 1989.
Pollard, Carl. 2005. *Hyperintensional semantics in a higher-order logic with definable subtypes*, Lambda Calculus, Type Theory, and Natural Language (Maribel Fernández, Chris Fox, and Shalom Lappin, eds.), King's College, London, 2005.
_____. 2008. *Hyperintensions*, Journal of Logic and Computation **18/2**, 257–282.

Potts, Christopher. 2002. *The lexical semantics of parenthetical* -as *and appositive* -which, Syntax **5/1**, 55–88.
Putnam, Hilary. 1958. *Formalization of the concept 'about'*, Philosophy of Science **25/2**, 125–130.
Rantala, Veikko. 1982. *Quantified modal logic*: *Non-normal worlds and propositional attitudes*, Studia Logica **41/1**, 41–65.
Reinhart, Tanya. 2006. *Interface Strategies*: *Optimal and costly computations*, Linguistic Inquiry Monographs, vol. 26, MIT Press, Cambridge, Mass.
Reynolds, John C. 1974. *Towards a theory of type structure*, Programming Symposium, Proceedings of 'Colloque sur la Programmation', 408–423.
Roelofsen, Floris. 2008. *Anaphora Resolved*, Dissertation, vol. 2008–09, Institute for Logic, Language and Computation, Amsterdam.
Rosetta, M. T. 1994. *Compositional Translation*, Kluwer International Series in Engineering and Computer Science, vol. 273, Kluwer Academic Publishers, Dordrecht.
Russell, Bertrand. 1905. *On denoting*, Mind **14/56**, 479–493.
Russell, Bertrand and Alfred North Whitehead. 1997. *Principia Mathematica to* ∗56, Cambridge Mathematical Library, Cambridge University Press, Cambridge.
Sag, Ivan, Gerald Gazdar, Thomas Wasow, and Steven Weisler. 1985. *Coordination and how to distinguish categories*, Natural Language and Linguistic Theory **3/2**, 117–171.
Sag, Ivan, Thomas Wasow, and Emily Bender. 2003. *Syntactic Theory*: *A formal approach*, CSLI Publications, Stanford.
Schönfinkel, Moses. 1924. *Über die Bausteine der mathematischen Logik*, Mathematische Annalen **92**, 305–316.
Scott, Dana. 1979. *Identity and existence in intuitionistic logic*, Applications of Sheaves (M. P. Fourman, C. S. Mulvey, and D. S. Scott, eds.), Lecture Notes in Mathematics, vol. 753, Springer, New York, 1979.
_____. 1980. *Relating theories of the λ-calculus*, To H. B. Curry: Essays on combinatory logic, lambda calculus and formalism (J. P. Seldin and J. R. Hindley, eds.), Academic Press, New York, 1980.
_____. 1982. *Domains for denotational semantics*, Automata, Languages and Programming: Lectures Notes in Computer Science **140**, 577–610.
Sheffer, Henry M. 1913. *A set of five independent postulates for Boolean algebras, with application to logical constants*, Transactions of the American Mathematical Society **14**, 481–488.
Simpson, Stephen G. 2009. *Subsystems of Second Order Arithmetic*, Perspectives in Logic, vol. 2, Cambridge University Press, Cambridge.
Smith, Jan. 1984. *An interpretation of Martin-Löf's type theory in a type-free theory of propositions*, Journal of Symbolic Logic **49/3**, 730–753.
Smullyan, Raymond M. 1995. *First-Order Logic*, Dover Publications, Mineola, NY.
Stainton, Robert J. 1996. *Philosophical Perspectives on Language*: *A concise anthology*, Mcgraw-Hill, Toronto.
_____. 2006. *Words and Thoughts*: *Subsentences, ellipsis and the philosophy of language*, Oxford University Press, Oxford.
Stalnaker, Robert. 1978. *Assertion*, Syntax and Semantics **9**, 315–332.
Steedman, Mark. 2012. *Taking Scope*: *The natural semantics of quantifiers*, MIT Press, Cambridge, Mass.
Stone, M. H. 1936. *The theory of representation for Boolean algebras*, Transactions of the American Mathematical Society **40/1**, 37–111.
Strawson, Peter F. 1950. *On referring*, Mind **59/235**, 320–344.

de Swart, Henriëtte. 1998. *Introduction to Natural Language Semantics*, CSLI Lecture Notes, CSLI Publications, Stanford.
Szabó, Zoltán Gendler. 2012. *Compositionality*, The Stanford Encyclopedia of Philosophy, Winter 2012 edition (Edward N. Zalta, ed.), 2012.
Tarski, Alfred. 1933. *The concept of truth in the languages of the deductive sciences*, Logic, Semantics, Metamathematics: Papers from 1923 to 1938 (John Corcoran, ed.), Hackett Publishing Company, Indianapolis, 1983.
_____. 1944. *The semantic conception of truth*, Philosophy and Phenomenological Research **4**, 341–375.
Thomason, Richmond H. 1980. *A model theory for the propositional attitudes*, Linguistics and Philosophy **4**, 47–70.
Tichý, Pavel. 1982. *Foundations of partial type theory*, Reports on Mathematical Logic **14**, 59–72.
Turner, Raymond. 1992. *Properties, propositions, and semantic theory*, Computational Linguistics and Formal Semantics (Michael Rosner and Roderick Johnson, eds.), Cambridge University Press, Cambridge, 1992.
_____. 1997. *Types*, Handbook of Logic and Language (Johan van Benthem and Alice G. B. ter Meulen, eds.), Elsevier Science Publishers, Amsterdam, 1997.
Veltman, Frank. 1981. *Data semantics*, Formal Methods in the Study of Language: Part 2 (Jeroen Groenendijk, Theo Janssen, and Martin Stokhof, eds.), Mathematical Centre Tracts, vol. 136, Mathematisch Centrum, Amsterdam, 1981.
_____. 1985. *Logics for Conditionals*, Dissertation, Jurriaans bv, Amsterdam.
_____. 1996. *Defaults in update semantics*, Journal of Philosophical Logic **25/3**, 221–261.
Wadsworth, Christopher P. 1976. *The relation between computational and denotational properties for Scott's D_∞-models of the lambda-calculus*, SIAM Journal on Computing **5/3**, 488–521.
Wansing, Heinrich and Nuel D. Belnap. 2010. *Generalized truth-values*: *A reply to Dubois*, Logic Journal of the IGPL **18/6**, 921–935.
Werning, Markus. 2005. *Right and wrong reasons for compositionality*, The Compositionality of Meaning and Content: Volume I: Foundational issues (Markus Werning, Edouard Machery, and Gerhard Schurz, eds.), Ontos Verlag, 2005.
Winter, Yoad. 1997. *Choice functions and the scopal semantics of indefinites*, Linguistics and Philosophy **20**, 399–467.
_____. 2004. *Functional quantification*, Research on Language and Computation **2**, 331–363.
Zaefferer, Dietmar. 2007. *Language as a mind sharing device*: *Mental and linguistic concepts in a general ontology of everyday life*, Ontolinguistics: How ontological status shapes the linguistic coding of concepts (Andrea C. Schalley and Dietmar Zaefferer, eds.), Trends in Linguistics, vol. 176, Mouton de Gruyter, Berlin, 2007.
Zalta, Edward N. 1983. *Abstract Objects*: *An introduction to axiomatic metaphysics*, Synthèse Library, vol. 160, D. Reidel, Dordrecht, Boston, and Lancaster.
_____. 1988. *Intensional Logic and the Metaphysics of Intentionality*, MIT Press, Cambridge, Mass.
_____. 1997. *A classically-based theory of impossible worlds*, Notre Dame Journal of Formal Logic **38/4**, 640–660.
_____. 1999a. *Natural numbers and natural cardinals as abstract objects*: *A partial reconstruction of Frege's* Grundgesetze *in Object Theory*, Journal of Philosophical Logic **28/06**, 617–658.
_____. 1999b. *Principia Metaphysica*, Manuscript, Stanford University.
Ziff, Paul. 1977. *About proper names*, Mind, New Series **86/343**, 319–332.

Summary

This dissertation is a contribution to the (onto-)logical foundations of natural language semantics. In particular, it provides a formal semantics for a canonical fragment of English which constructs the referents of the fragment's expressions from a *single* basic type of object (rather than from two or more basic types, as is standard in Montague semantics). The possibility of such a *single-type semantics* has been conjectured by Partee (**2006**) to account for recent findings in language development. The presented semantics supports Partee's conjecture and accommodates its empirical motivation. The development of this semantics answers a number of questions about the salience of the Montagovian type system, the robustness of semantic theories with respect to their basic-type choice, and their classification according to their objects' informational strength.

To give the idea of a single-type semantics intuitive content, the first part of this dissertation presents the simplest single-type semantics which supports Partee's conjecture. This semantics is a version of Henkin's Theory of Propositional Types that interprets all expressions from the fragment into constructions out of the basic type o. As a result of the neutrality of the type o between Montague's basic types for individuals (type e) and propositions (type $\langle s,t\rangle$), proper names (traditionally, expressions of the type e) and sentences (traditionally, expressions of the type $\langle s,t\rangle$) will both receive an interpretation in this type. However, as a result of the basic type's primitiveness, the theory lacks a truth-definition and fails to instantiate some semantic consequences of Partee's conjecture (e.g. the existence of equivalence relations between names and sentences). Thus, our o-based semantics disqualifies as a suitable single-type semantics for the fragment.

The remainder of the dissertation develops a single-type theory which accommodates the consequences of Partee's conjecture. To this aim, Part II identifies two suitable Montague types which analyze the type o from Part I. Part III defines the associated semantics of these two types.

Specifically, to find the 'best' single-type candidate, Part II introduces a set of semantic requirements (including algebraicity and partiality) which ensure the type's suitability as a single semantic basis for the fragment. The application of these requirements to the set of Montague types identifies the types $\langle s,t\rangle$ (for propo-

sitions) and $\langle s, \langle s, t\rangle\rangle$ (for propositional concepts) as the best-suited candidates. The latter are shown to have desirable meta-properties. Thus, they are equivalent (up to coding) to their representations in higher-rank types, and are associated with differently informative semantic theories. As a result of the former, the provision of a single-type semantics is independent of a particular basic-type choice. As a result of the latter, single-type theories are classified by a natural ordering relation.

Part III provides a detailed presentation of our $\langle s, t\rangle$- and $\langle s, \langle s, t\rangle\rangle$-based single-type semantics, in which we distinguish two classes ('weak' and 'strong'). Since propositional concepts represent individuals via functions from indices to the intersection of all true propositions about the individuals at those indices, we describe our $\langle s, \langle s, t\rangle\rangle$-based single-type semantics as a *strong single-type semantics*. The semantics of the type $\langle s, t\rangle$, which only represents individuals via the set of indices in which they exist, is described as a *weak single-type semantics*. We adapt all relevant concepts of the o-based logic from Part I to constructions out of the type $\langle s, t\rangle$, resp. $\langle s, \langle s, t\rangle\rangle$ and formulate a set of semantic constraints for non-logical single-type constants. Such constraints ensure the interpretation of proper names and sentences into objects of the form from Part II. We show that these constraints enable the identification of equivalence relations between proper names and sentences.

The conclusion locates the dissertation in the wider context of formal semantics. In particular, it presents three precursors of single-type semantics (i.e. Kratzer-style Situation Semantics, Zalta's Abstract Object Theory, and Chierchia and Turner's Property Theory), sketches alternative approaches to the provision of a single-type semantics, and identifies new areas for future work. Specifically, the representational strategies of the single-type semantics from Part III formalize Kratzer's situation-theoretic interpretation of names, and improve upon the property-theoretic type system of Chierchia and Turner. Future work includes the provision of a type-shifting account of nonsentential assertion (along the lines of Part III), an extension of our single-type semantics to definite and indefinite noun phrases, and the use of this semantics in the unification of the semantic ontologies of larger fragments of natural language.

Samenvatting

Dit proefschrift is een bijdrage tot de (onto-)logische grondslagen van de semantiek van natuurlijke taal. In het bijzonder ontwikkelt de thesis een formele semantiek voor een fragment van het hedendaags Engels, waarin de betekenissen van linguïstische uitdrukkingen opgebouwd worden uit *één* basistype. (De klassieke formele semantiek, cf. (**Montague, 1970a**; **1973**), onderstelt ten minste twee verschillende basistypes). De mogelijkheid voor een dergelijke *enkele-type-semantiek* [Engels *single-type semantics*] werd in 2006 door Barbara Partee voorgestelt om recente observaties in de taalontwikkeling te modelleren. De semantiek die in dit werk beschreven wordt, bewijst Partee's vermoeden, en biedt een kader voor de genoemde empirische feiten. De ontwikkeling van een enkele-type-semantiek beantwoordt een aantal vragen in verband met de rol van Montague's type-systeem, de robuustheid van semantische theoriën wat betreft de keuze van basistypes, en hun (zelf-)classificatie volgens de informatie-inhoud van hun objecten.

Om de lezer een intuïtief idee te geven van de enkele-type-semantiek, beschouwen we in het eerste deel van dit proefschrift de eenvoudigste enkele-type-semantiek die Partee's vermoeden bevestigt. Deze semantiek is een versie van Henkin's Theorie der Propositionele Types, die alle uitdrukkingen uit het fragment als constructies uit de basistype o vertaalt. Als gevolg van de neutraliteit van het type o tussen Montague's basistypes voor individuen (type e) en proposities (type $\langle s, t\rangle$) verkrijgen zowel eigennamen (normalerwijze type e) als zinnen (normalerwijze type $\langle s, t\rangle$) een interpretatie in dit type. Echter, als gevolg van het primitieve karakter van de basistype o heeft de theorie geen waarheidsdefinitie, en valideert sommige semantische gevolgen van Partee's vermoeden (bijv. het bestaan van equivalentie-relaties tussen eigennamen en zinnen) niet. Het weze duidelijk dat deze 'primitieve aanpak' niet resulteert in een bruikbare enkele-type-semantiek en een fundamentele verfijning noopt zich. Het leeuwendeel van dit werk bestaat erin deze verfijning van *naaldje tot draadje* te beschrijven in Delen II en III.

Aldus, in Delen II en III van dit proefschrift wordt een enkele-type-semantiek ontwikkelt die de semantische gevolgen van Partee's vermoeden valideert. In Deel II identificeren we twee geschikte Montague-types die de rol van de type o uit Deel I spelen, en in Deel III beschrijven we een semantiek gebaseerd op deze types.

Om de beste kandidaat voor het basistype te identificeren, introduceren we in Deel II een reeks van semantische eigenschappen (waaronder algebraiciteit en partialiteit) die de geschiktheid van het basistype garanderen. De toepassing van deze criteria op de verzameling van Montague-types identificeert de types $\langle s, t\rangle$ (voor proposities) en $\langle s, \langle s, t\rangle\rangle$ (voor propositionele concepten [Engels *propositional concepts*]) als de beste enkele-type-kandidaten. Deze laatste hebben uiterst wenselijke meta-eigenschappen: Zo zijn de genoemde basistypes equivalent met hun representaties in hogere-rang types (op wat codering na), en zijn geassocieerd met semantische theoriën met verschillende informatieve inhoud. Als gevolg van deze eigenschappen is de enkele-type-semantiek onafhankelijk van de specifieke keuze van een (rijk genoeg) basisobject. Ook geeft deze observatie aanleding tot een classificatie van alle mogelijke enkele-type-semantieken volgens een natuurlijke orderingsrelatie gebaseerd op de complexiteit van het basistype.

In Deel III bevindt zich een gedetailleerde beschrijving van de $\langle s, t\rangle$- en $\langle s, \langle s, t\rangle\rangle$-gebaseerde enkele-type-semantieken waarin we twee categorieën ('sterk' en 'zwak') onderscheiden. Aangezien propositionele concepten individuen e voorstellen als functies van indices naar de doorsnede van alle ware proposities over deze individuen in deze indices, spreken we in het geval van de basistype $\langle s, \langle s, t\rangle\rangle$ van een *sterke enkele-type-semantiek*. Omdat het type $\langle s, t\rangle$ individuen enkel via de verzameling van indices vorstelt waar ze bestaan, spreken we in het geval van de type $\langle s, t\rangle$ van een *zwakke enkele-type-semantiek*. Daarnaast gaan we dieper in op de vertaling van alle relevante concepten uit de o-gebaseerde logica uit Deel I naar constructies uit de types $\langle s, t\rangle$ en $\langle s, \langle s, t\rangle\rangle$. Ten laatste bespreken we de specificatie van een collectie van semantische randvoorwaarden voor de niet-logische enkele-type-constanten. Deze randvoorwaarden verzekeren de interpretatie van eigennamen en zinnen naar objecten zoals ge-introduceerd in Deel II, en maken de identificatie van equivalentie-relaties tussen eigennamen en zinnen mogelijk.

In de conclusie wordt dit proefschrift geduid in de ruimere context van de formele semantiek. We presenteren verbindingen met drie voorgangers van de enkele-type-semantiek (i.e. Kratzer's Situatiesemantiek, Zalta's Abstracte-Objectentheorie en Chierchia en Turner's Eigenschappentheorie), schetsen alternatieve formuleringen van de enkele-type-semantiek, en identificeren nieuwe vruchtbare gebieden voor toekomstig onderzoek. We merken op dat de representationele strategieën van de enkele-type semantiek aanleiding geven tot een formalisatie van Kratzer's situatie-theoretische interpretatie van eigennamen, en tot een verbetering van het eigenschap-theoretische type-systeem van Chierchia en Turner. Toekomstig werk omvat de behandeling van type-schakeling voor niet-zinsconstructies (als in Deel III), een uitbreiding van de beschreven enkele-type-semantiek naar definiete en niet-definiete NPs, en het gebruik ervan in de unificatie van ontologieën van grotere delen van de natuurlijke taal.

Zusammenfassung

Diese Dissertation liefert einen Beitrag zu den (onto-)logischen Grundlagen der natürlichsprachigen Semantik. Im Besonderen präsentiert die Arbeit eine formale Semantik für ein Standard-Fragment des Englischen, die die Bezugsgegenstände sprachlicher Ausdrücke aus nur *einem* Basistyp konstruiert. (In der klassischen formalen Semantik, siehe (**Montague, 1970a**; **1973**), werden gewöhnlich mindestens zwei Basistypen angenommen). Die Möglichkeit einer solchen *ein-getypten Semantik* [Engl. *single-type semantics*] wurde 2006 von Partee vorgeschlagen, um neue Forschungsergebnisse aus der Sprachentwicklung zu modellieren. Die neue Semantik stützt Partees Hypothese und trägt ihrer empirischen Motivation Rechnung. Die Entwicklung der Semantik beantwortet eine Reihe von Fragen über die Motivation des Montagovischen Typensystems, die Robustheit von semantischen Theorien in Bezug auf ihre Typenwahl, und ihre Klassifizierung entsprechend des Informationsgehaltes ihrer Gegenstände.

Um dem Leser den Einstieg in die ein-getypte Semantik zu erleichtern, präsentiert der erste Teil der Dissertation die einfachste ein-getypte Semantik, die Partees Hypothese stützt. Letztere ist eine Variante von Henkins Theorie Propositionaler Typen, die alle Ausdrücke des Fragments als Konstruktionen aus dem Basistyp o interpretiert. Aufgrund der Neutralität des Typs o zwischen Montagues Basistypen für Individuen (Typ e) und Aussagen (Typ $\langle s,t\rangle$) werden Eigennamen (traditionell, Ausdrücke des Typs e) und Aussagesätze (traditionell, Ausdrücke des Typs $\langle s,t\rangle$) beide in diesem Typ interpretiert. Infolge der Primitivität des Basistyps o verfügt die Theorie indes über keine einfache Wahrheitsdefinition, und vermag einige semantische Konsequenzen von Partees Hypothese (z.B. die Existenz von Äquivalenzbeziehungen zwischen Eigennamen und Sätzen) nicht zu instanziieren. Somit scheitert die o-basierte Semantik als eine geeignete ein-getypte Semantik.

Der Rest der Dissertation entwickelt eine ein-getypte Theorie, die die Konsequenzen von Partees Hypothese erfasst. Hierzu identifiziert Teil II zwei geeignete Montague-Typen, die den Typ o aus Teil I interpretieren. Teil III definiert die Semantik dieser beiden Typen.

Um den 'besten' Kandidaten für den Basistyp einer ein-getypten Semantik

zu finden, führt Teil II eine Menge von semantischen Voraussetzungen ein, die die Eignung des Typs als eine semantische Basis für das Sprachfragment gewährleisten. Die Anwendung dieser Voraussetzungen auf die Menge von Montague-Typen identifiziert die Typen $\langle s, t\rangle$ (für Aussagen) und $\langle s, \langle s, t\rangle\rangle$ (für Aussagenkonzepte [Engl. *propositional concepts*]) als die bestgeeigneten Kandidaten. Letztere haben wünschenswerte Metaeigenschaften. So sind sie mit ihren Repräsentationen in höhergeordneten Typen bis zur Codierung äquivalent und werden mit verschieden informativen semantischen Theorien verbunden. Aufgrund der ersten Eigenschaft ist die Formulierung einer ein-getypten Semantik von der Wahl des Basistyps unabhängig. Aufgrund der zweiten Eigenschaft werden ein-getypte Theorien durch eine natürliche Ordnungsrelation klassifiziert.

Teil III liefert eine ausführliche Darstellung der $\langle s, t\rangle$- und der $\langle s, \langle s, t\rangle\rangle$-basierten ein-getypten Semantiken, in der wir zwei verschiedene Klassen ('stark' und 'schwach') unterscheiden. Da Aussagenkonzepte Individuen durch Funktionen von Indizes (möglichen Situationen) zum Durchschnitt aller wahren Aussagen über die Individuen in diesen Indizes repräsentieren, beschreiben wir die $\langle s, \langle s, t\rangle\rangle$-basierte ein-getypte Semantik als eine *starke ein-getypte Semantik*. Die Semantik des Typs $\langle s, t\rangle$, die Individuen nur durch die Menge der Indizes repräsentiert, in denen sie existieren, wird als eine *schwache ein-getypte Semantik* beschrieben. Wir adaptieren alle relevanten Konzepte aus der o-basierten Logik von Teil I an Konstruktionen aus dem Typ $\langle s, t\rangle$ bzw. $\langle s, \langle s, t\rangle\rangle$ und formulieren eine Klasse von semantischen Einschränkungen für nicht-logische ein-getypte Konstanten. Solche Einschränkungen gewährleisten die Interpretation von Eigennamen und Sätzen in Gegenstände der Form von Teil II. Wir zeigen, dass unsere Definitionen die Identifizierung von Äquivalenzbeziehungen zwischen Eigennamen und Sätzen ermöglichen.

Das Schlusskapitel stellt die vorliegende Dissertation in den weiteren Kontext der formalen Semantik. Insbesondere präsentiert es drei Vorläufer der ein-getypten Semantik (d.s. Kratzers Situationen-Semantik, Zaltas Theorie Abstrakter Gegenstände und Chierchia und Turners Eigenschaftstheorie), skizziert alternative Ansätze für eine ein-getypte Semantik und identifiziert zukünftige Arbeitsbereiche. Insbesondere formalisieren die Repräsentationsstrategien der ein-getypten Semantik aus Teil III Kratzers Situationen-theoretische Interpretation von Eigennamen und verbessern das eigenschaftstheoretische Typensystem von Chierchia und Turner. Zukünftige Arbeit wird sich auf typenschaltende Ansätze nicht-sententieller Behauptung (analog zu Teil III), die Erweiterung unserer ein-getypten Semantik auf definite und indefinite Nominalphrasen und ihre Verwendung in der Vereinheitlichung größerer Sprachfragmente konzentrieren.

Abstract

In (**Montague, 1970a**), Montague defines a formal theory of linguistic meaning which interprets a small fragment of English through the use of *two* basic types of objects: individuals and propositions. In this dissertation, we develop a comparable semantics which only uses *one* basic type of object (hence, *single-type semantics*).[4] Such a semantics has been conjectured by Partee (**2006**) as a 'minimality test' for the Montagovian type system, which captures the lowest ontological requirements on any successful semantics for Montague's fragment. The resulting semantics unifies the semantic ontology of the fragment and yields insight into the apparatus of types in formal semantics.

To give the idea of a single-type semantics intuitive content, the first part of this dissertation identifies a simple single-type semantics whose basic type is neutral between individuals and propositions. However, as a result of the type's primitiveness, the theory lacks a number of core semantic notions (like truth and (certain cases of) equivalence). To compensate for this shortcoming, Part II identifies two Montague types which analyze the basic type from Part I. Part III defines the single-type semantics which are associated with these two types. These semantics are models of subsystems of a variant of Montague's Intensional Logic. Their interpretation of Montague's fragment identifies the constraints on any 'usable' single-type theory.

The semantics from Part III are shown to fit squarely with Kratzer-style Situation Semantics and to improve upon the property-theoretic type system of Chierchia and Turner.

Keywords Single-type hypothesis, Montague semantics, Natural language metaphysics, Type theory, Unification.

[4]This abstract is intended for formal semanticists, logicians, and philosophers of linguistics. The Preface and Chapter 1 provide a more gentle introduction to single-type semantics.

Curriculum Vitæ

Kristina Liefke was born in Neumünster, Germany, on April 20, 1983. After spending her junior year of high school in Fort Worth, Texas and graduating with honors from Klaus-Groth-Gymnasium (Neumünster), she took up the study of philosophy and English linguistics at Christian-Albrechts-University (CAU) at Kiel, Germany, and at UCLA. In 2009, she completed her studies at CAU with a Master's thesis on Frege's notion of function (written during a DAAD-sponsored stay at the Harvard and UMass Amherst Linguistics Departments). Since 2009, she has been a PhD student at the Tilburg Center for Logic and Philosophy of Science (Tilburg University). In 2012, she moved to the Munich Center for Mathematical Philosophy (Ludwig-Maximilians-University Munich, Germany). Kristina is married to the mathematician Sam Sanders.